University of Southern Queensland

Faculty of Health, Engineering & Sciences

Signal Processing Techniques for Machine Condition Monitoring

A dissertation submitted by

J. Steele

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Electrical & Electronic Engineering

Submitted: October, 2016

Abstract

The purpose of this dissertation is to analyse and compare a wide range of wavelet de-noising parameters and determine which parameters are best suited to the de-noising of rolling element bearing vibration signals.

The condition of rolling element bearings is often monitored by recording the vibration of the bearing using accelerometers and analysing the signal for particular frequency content. When a bearing experiences a mechanical fault the vibration signal will contain frequencies relating to the failing component. Monitoring the bearing vibration can provide advanced warning that a bearing failure is imminent. However, when a mechanical fault is in the early stages of development the fault frequency can be very low in magnitude and difficult to detect. Improving the signal to noise ratio of these fault frequencies can provide earlier detection of the fault.

One method to improve the signal to noise ratio of a bearing fault frequency is to reduce the noise component in the vibration signal using wavelet theory. Wavelet de-noising has many parameters that can be varied which changes how the de-noising process modifies the vibration signal in the time domain. This dissertation makes comparison between the many de-noising parameters available and assesses which parameters provide the best increase in signal to noise ratio.

The wavelet de-noising process alone does not identify the frequencies relating to a bearing fault. The frequency content within the vibration time domain signal is required to be extracted and assessed to determine the effect of the wavelet de-noising process. Three frequency extraction methods were used to analyse the de-noised signals and indicate the magnitude of signal to noise ratio improvement achieved through de-noising. This dissertation shows that cepstrum analysis of the time domain signal responded best to the wavelet de-noising process and large improvements in signal to noise ratio were realised.

University of Southern Queensland

Faculty of Health, Engineering & Sciences

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean

Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

J. Steele

0050104035

Acknowledgments

First and foremost I would like to thank my wife Ellen for her kindness and patience during the last five years.

I would also like to thank my supervisor John Leis. His lectures and further discussions have inspired me to pursue signal processing.

J. Steele

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rolling Element Bearing Condition Monitoring

Condition monitoring is the practice of observing and monitoring particular machine parameters using real time and historical data, in order to determine the current condition and future maintenance requirements of the machine.

Many dozens of methods exist to monitor a machine's condition. Vibration, noise, acoustic emission, electrical signatures (voltage and current), thermography and imaging, and wear particle analysis are the broad headings that encompass the many options for condition monitoring. Condition monitoring techniques are most commonly applied to rotating machinery and their components, such as bearings and gears.

This body of work focuses on signal processing and frequency analysis applications for condition monitoring of rolling element bearings. Signal processing of bearing parameters such as vibration, provide real time feedback of the current wear condition of the bearing. When correctly applied, signal processing methods can indicate precise bearing fault information including the type and location of fault or faults. A common problem that impedes this task is background signal noise reducing the signal to noise ratio and masking the signature of a bearing fault. This work will investigate and compare noise removal options that make use of wavelet theory.

1.2 Signal Noise Removal using Wavelet Theory

Traditional techniques for removing noise from a time domain signal involve a filter or combination of filters. The time domain signal is passed through the filter where unwanted bands of frequencies, such as high frequency noise components are blocked, and desired frequencies are permitted to pass.

Removing signal noise using wavelet theory takes an alternative approach, making use of the wavelet transform where the time domain signal is transformed into the time-frequency domain. The time-frequency domain provides information on what frequencies are present and when they are occurring in the time domain signal. By assessing the magnitude of each frequency component in the time-frequency domain the analyst can make a statistical estimation as to whether the frequency component forms part of the desired signal, or is contributing to signal noise.

Wavelet de-noising theory is centred around statistical analysis of the time domain signal in order to set thresholds that define whether a frequency component is classed as signal or noise. The frequency components deemed to be noise are removed from the time-frequency domain, and the remaining components are deemed to be part of the desired signal. Using the remaining frequency components, and the knowledge of when they occur in the time domain signal, the time domain signal can be reconstructed minus the noise components. The original signal is de-noised.

1.3 Aims of this work

Early detection of a failure mode in a machine is critical for providing safe machine operation and lower maintenance costs by allowing planning of future maintenance requirements. Detecting a fault in a machine is made difficult in a field environment due to signal interference and noise issues. Often the fault must increase in severity and overcome background noise in order to be detected. Broadly speaking this body of work aims to provide earlier detection of machine faults by improving signal to noise ratio of a fault signal.

This work focuses on the use of wavelets to remove noise from a bearing vibration signal.

Many wavelet de-noising parameters are available for use and this work aims to make comparison between the wide range of options available, and determine optimal parameters when de-noising bearing vibration signals.

Similar studies have been completed in the past where a selection of de-noising methods are compared using a single set of simulated data or data from a controlled testbed arrangement. Other studies compare a small selection of de-noising methods against a broader range of signals. This study combines both concepts to compare a broad range wavelet de-noising methods against a broad range of simulated data, short duration testbed data, and longer duration datasets that track bearing wear over a period of days.

1.4 Overview of the Dissertation

This dissertation is organized as follows:

Chapter 2 Literature Review.

Chapter 3 Implementation in MATLAB^{TM}.

Chapter 4 Signals.

Chapter 5 Methodology.

Chapter 6 Results.

Chapter 7 Analysis.

Chapter 8 Conclusion.

Chapter 2

Literature Review

2.1 Chapter Overview

Vibration signature analysis identifies fault features within the vibration signal of a machine or structure. The literature shows vibration analysis is mostly applied to rotating machinery containing gears and bearings, structures which require closer monitoring of their condition, such as wind turbine blades, and machine cutting tools for lathes and mills. The following literature review focuses on bearing vibration.

2.2 Time Domain

The literature reveals many techniques for analysing bearing vibration in the time domain. The review paper by Mathew, Patil & RajendraKumar (2008) explains the time domain approach involves inspecting the historical vibration data signal and extracting time waveform indices, probability density functions and probability density moments. They say the probability density moments provide the most detail as the odd moments indicate the peak moments with respect to the mean, and the even moments are proportional to the spread of the distribution. In Chapter 25 of Vibration and Shock Handbook, Mechefske (2005) also recommends the use of probability density moments.

Other statistical parameters are used to analyse data in the time domain. Akturk & Karacay (2009) explain the use of peak-to-peak amplitude history measurement, Root-

Mean-Square (RMS) history measurement, crest factor and kurtosis as scalar indicators to define a level of damage in a bearing. Bolaers, Dron & Rasolofondraibe (2004) demonstrate the effectiveness of kurtosis after pre-filtering and selecting the frequency band for a given bearing characteristic frequency fault. Al-Balushi & Samanta (2003) combine the more common statistical parameters such as RMS, skewness (an odd probability density moment) and kurtosis with an artificial neural network (ANN) algorithm to diagnose bearing condition.

Although the use of statistical parameters is effective in characterising the energy in the signal of the damaged bearing, such methods are not able to specify the position or nature of the defects (Akturk & Karacay 2009). Mathew et al. (2008) explain these methods can only be applied in the early stages of a fault. As the fault worsens the vibration signal becomes more random and the noise floor increases in amplitude. The limits of crest factor and kurtosis are highlighted by Fray, Pachaud & Salvetat (1997) and Mathew et al. (2008) who support the claim that these indicators are only useful for characterising the energy in the signal.

2.3 Frequency Domain

The literature reveals frequency domain analysis is the most common and arguably the most effective technique for detecting fault signatures within a bearing vibration signal. Kiral & Karagülle (2003) says the presence of a defect frequency in the frequency domain is a powerful indication of a fault with the bearing, and Choi, Kim & Park (2013) makes note that the frequency domain parameters are generally more consistent than the time domain parameters.

Typically a bearing is constructed of four components. The inner race, the outer race, the cage and the rollers (note that many rollers may feature in a particular bearing design, but for this analysis their system is considered as one component). Localised faults consist of cracking of races or cage, pits and spalling and the dominant fault mode is spalling of either the inner or outer race (Choi et al. 2013). Early research in this field by Gustafsson & Tallian (1962) and Harris (1966) studied the frequency response when each of these bearing components experiences a single fault. This resulted in the creation of four bearing characteristic fault frequencies (CFF). The following characteristic

frequencies are for bearings with a stationary outer race (Choudry & Tandon 1999). They do make note that the characteristic fault frequencies may differ slightly in practice than the calculated frequencies due to roller slipping and skidding.

Outer Race Defect Frequency,

$$\omega_{od} = \frac{Z\omega_s}{2}\left(1 - \frac{d}{D}\cos\alpha\right) \tag{2.1}$$

Inner Race Defect Frequency,

$$\omega_{id} = \frac{Z\omega_s}{2}\left(1 + \frac{d}{D}\cos\alpha\right) \tag{2.2}$$

Ball Spinning Frequency,

$$\omega_b = \frac{D\omega_s}{2d}\left[\left(1 - \frac{d}{D}\cos\alpha\right)^2\right] \tag{2.3}$$

and Cage Frequency,

$$\omega_c = \frac{\omega_s}{2}\left(1 - \frac{d}{D}\cos\alpha\right) \tag{2.4}$$

where ω_s is the shaft rotation frequency in radians/second, d is the diameter of the rolling element, D is the pitch diameter, Z is the number of rolling elements and α is the contact angle.

The literature shows that the majority of frequency analysis techniques for detecting faulty bearings primarily search for one of the above four frequencies within the vibration signal.

Arniaz, Bediaga, Mendizabal & Muñoa (2013), Kiral & Karagülle (2003) and Dowling (1993) discuss the application of standard spectral analysis using the Fast Fourier Transform (FFT) of the vibration signal. Arniaz et al. (2013) informs the vibration signal generated by a bearing with a fault produces significant peaks at the frequency determined by one of the above five equations. Figure 2.1 shows the frequency spectrum of a vibration signal of a bearing with an outer race fault. The peaks of the fault frequency can be clearly seen.

The major flaw with using simple spectrum analysis is that a fault in the bearing component must create a vibration with sufficient amplitude to be detected. Kiral & Karagülle (2003) and Arniaz et al. (2013) report that faults at their early stages can be difficult

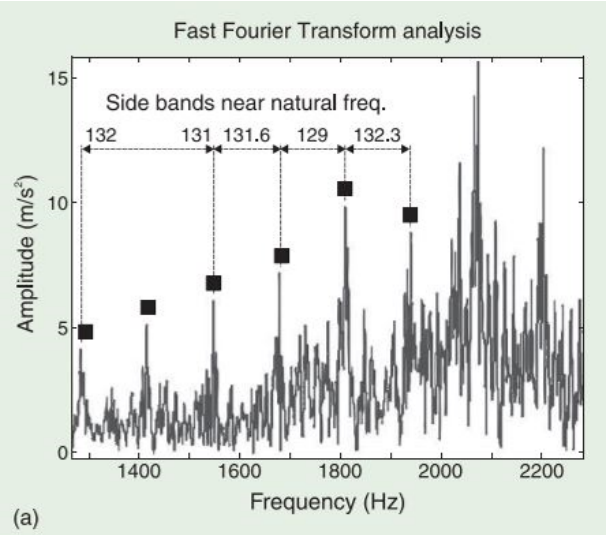

Figure 2.1: A Fast Fourier Transform analysis spectrum for a bearing with an outer race fault (Arniaz et al. 2013).

to detect because the low amplitude of the fault frequencies can be drowned out by the resonant amplitudes of other components. Dowling (1993) says each bearing straight off the assembly line already has some form of characteristic fault frequency due to manufacturing process. He also highlights the point that the bandwidth of the vibration spectrum is quite narrow and susceptible to noise in industrial environments.

The literature also indicates the natural frequency envelope technique as a successful method of detecting bearing faults. Dowling (1993) explains that a faulted component on the bearing creates Dirac delta type impulses which excite the natural frequency of the bearing structure.

The natural frequency of the structure is usually much higher than the vibration bandwidth, often more than 5kHz (Choudry & Tandon 1999). This higher frequency can be thought of as a carrier frequency onto which a signal can be modulated, whereas the modulating signal is the bearing characteristic frequency (Habetler, Harley & Stack 2004). The method involves analysing the natural frequency transients caused by the bearing fault, in particular the frequency content of the transients, in order to detect the bearing characteristic frequencies (Dowling 1993). According to Dowling (1993) and Choudry & Tandon (1999) this concept has other names such as High Frequency Resonance, Stress Wave Analysis and Shock Pulse Analysis.

Dowling (1993), Kiral & Karagülle (2003) and Arniaz et al. (2013) describe the same

algorithm for the process. To begin, the natural frequency of the bearing must be identified. This can be achieved by either converting the vibration signal to frequency domain using FFT and observing the concentration of frequencies around a peak above 5 kHz, or calculating the bearing natural frequency using Finite Element Analysis (FEA). The next step is to band-pass filter the vibration signal around the natural frequency identified. This isolates the frequency components which make up the natural frequency transient. Within this information is the fault frequency but further analysis is required. The envelope of the filtered signal is extracted using either amplitude demodulation or a Hilbert transform. The result is low frequency variations of the high frequency signal. The frequency spectrum of the envelope is then extracted using FFT where characteristic fault frequencies can be identified.

Dowling (1993) claims the natural frequency envelope technique is more sensitive for detecting impulsive events, but the method does have its drawbacks. The bearing natural frequency must be known *a priori*, or at least must be determined during the processing, making the automatic selection of the band-pass filter limits challenging. Any changes to the structure inherently changes the natural frequency of the system which (Kiral & Karagülle 2003). Such changes can result from methods of mounting the bearing or even different positions the machine housing the bearing may be in.

Another approach to solving the shortcomings of the standard spectral analysis of vibration signals led to the application of cepstrum (or cepstral) analysis. Cepstrum analysis is essentially analysing the spectrum of the logarithm of the vibration signal spectrum and can be calculated as a real power spectrum or complex spectrum (Dowling 1993). Originally created by Bogert, Healy & Tukey (1963) to study seismic echoes in the ground, cepstrum analysis is used to observe periodic impulses in a vibration signal (Choi et al. 2013). The power cepstrum is commonly used in vibration analysis and has been further developed into other techniques to improve its noise immunity such as the minimum variance cepstrum (Choi et al. 2013).

el Badaoui, Daniére & Guillet (2004) explains that a delayed echo in a signal manifests itself as a ripple in the logarithmic spectrum. The echo within the signal is one of the characteristic frequencies mentioned above. The frequency of the ripple is determined by calculating the spectrum of the logarithmic spectrum and will be represented as a peak. It is important to note the units along the x axis in this result are in time and not

frequency. The power cepstrum is defined by el Badaoui et al. (2004) as,

$$power\,cepstrum\,of\,signal = \left| F^{-1}\left[log\Big(\left| F(f(t)) \right|^2 \Big) \right] \right|^2 \tag{2.5}$$

where F stands for Fourier Transform.

A problem identified by Dowling (1993) is that cepstrum analysis is particularly sensitive to characteristic fault frequency harmonic activity which commonly occurs in degrading machines. If such spacing of harmonic components can be associated with a fault then the technique is quite successful but as Arniaz et al. (2013) indicates, if the spacing frequencies of the harmonics are not uniform then cepstrum analysis can suffer.

The frequency domain methods described above for extracting the characteristic fault frequencies all share common problems. First of all, the fault signature must be periodic in the signal however during the early stages of a fault the fault signatures are non-periodic (Dowling 1993).Dowling (1993) also says that by the time the fault signature has become periodic the fault itself is quite well developed. The second common problem is that the methods described above are only suitable for stationary signals however many bearing faults provide a non-stationary fault signal. The final problem that affects the success of the above described methods is that of noise. Background bearing vibrations such as cage noise, squeal noise, race noise and noise resulting from manufacturing flaws (Momono & Noda 1999) all act to create a noisy environment where the characteristic fault frequencies have a low SNR.

2.4 Time-Frequency Domain

As discussed above, frequency domain analysis is suited for analysing vibrations signals that are periodic with few or no transients. However, transient behaviour can result from speed and load variations (Kalista & Liska 2015) and even electric induction motors have small speed oscillations as a result of the 3 phases and supply frequency. Due to the complex and dynamic nature of vibration signals, which often contain transient and non-stationary components, time-frequency analysis techniques have been developed to optimise the resolution in the time and frequency domain simultaneously (Kostopouolos & Loutas 2012).

Many dozens of different methods have been found in the literature review. These cover areas such as training artificial neural networks, detecting singularities within signals and mapping phase changes of fault parameters. The following review of the literature focuses on using time-frequency analysis to detect the bearing characteristic fault frequencies.

A technique called Short Time Fourier Transform (STFT) is one of the more basic time-frequency analysis techniques applied to vibration signal analysis. The concept involves dividing a non-stationary signal into sections where stationary assumptions may apply, and then conduct a Fourier transform to each section (Gade & Gram-Hansen 1996). By analysing each section individually the frequency content of each section in time can be extracted. The length of the section (often referred to as a window) can be arbitrarily set to any length, however the following must be considered; a long window provides higher frequency resolution but lower time resolution, whereas a short window provides high time resolutions but lower frequency resolution (Kalista & Liska 2015).

The equation of the STFT is as follows,

$$S_b = \int_{-\infty}^{\infty} x(t)g^*(t-b)\mathrm{e}^{-j2\pi\omega(t-b)}dt \tag{2.6}$$

where *x(t)* is the time signal and b is the various positions of the window.

The product of the STFT is a 3 dimensional plot. The X axis is time, the Y axis is frequency and the Z axis is the strength of a frequency component at that moment in time. Figure 2.2 shows a STFT plot of a healthy bearing vibration signal. Figure 2.3 shows a STFT plot of a bearing signal containing a fault which shows a repeating signal at approximately 300Hz. The repeating signal is one of the characteristic fault frequencies for this bearing.

Despite the simplicity of the STFT technique it does not appear to be widely used as a method of time-frequency analysis Chandra & Sekhar (2016) and Al-Badour, Cheded & Sunar (2010) used STFT successfully to detect bearing defects however their work involved comparing STFT against other time-frequency analysis techniques. Weaknesses of the STFT are the compromise between time and frequency resolution, and the fact that the resolution is constant throughout the analysis. If higher time frequency is required then lower frequency content cannot be extracted, and vice versa.

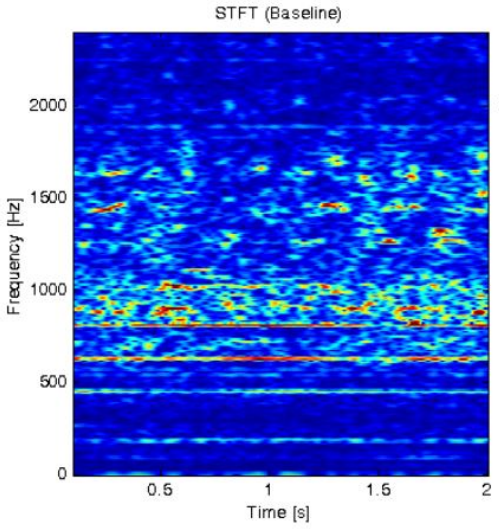

Figure 2.2: A Short Time Fourier Transform plot of a healthy bearing (Gade & Gram-Hansen 1996).

An improvement on the STFT is the wavelet transform. In a similar fashion to the STFT the wavelet transform can measure time-frequency variations of spectral components within a signal however it has the advantage of being able to use varying time and frequency resolutions (Mallat 1998).

A wavelet is simply a normalised function with zero average that is dilated with a scale parameter and translated with a translation parameter. Many varieties of wavelets exist. Figure 2.4 shows a Mexican Hat wavelet whose function is defined by,

$$\psi(t) = \frac{2}{\pi^{1/4}\sqrt{3\sigma}}\left(\frac{t^2}{\sigma^2} - 1\right)\mathrm{e}^{\frac{-t^2}{2\sigma^2}} \tag{2.7}$$

Some other wavelet bases are shown in Figure 2.5.

The scale parameter stretches or compresses the wavelet while the translation parameter shifts the location of the wavelet up or down the time series as required. The wavelet can be thought of as a band pass filter centred on a frequency f0. This centre frequency is the reciprocal of the time period of the wavelet (Gade & Gram-Hansen 1996). As the frequency scale increases the time period compresses in proportion improving the time resolution, and vice versa.

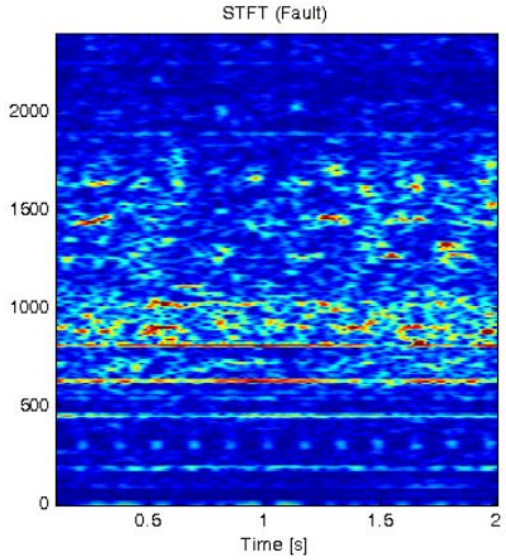

Figure 2.3: A Short Time Fourier Transform plot of a faulty bearing. (Gade & Gram-Hansen 1996).

A wavelet transform correlates a wavelet with a time series signal and then takes the integral of the product. A wavelet transform conducted just once produces a single wavelet coefficient, also known as a wavelet atom. The wavelet atom is a scalar representation of the energy contained within the frequency scale (the dilation) and time (translation) selected. The wavelet atom can symbolically be represented by a rectangle on a time-frequency plane as shown in Figure 2.6.

Families of wavelet atoms representing different time and frequency content are created by dilating and translating the wavelet and then calculating the wavelet transform. The wavelet atoms can then be arranged on a two dimensional plot using time and frequency as the X and Y axes respectively. Equation 2.8 shows how a typical wavelet is dilated by s and translated by u, where ψ is a wavelet function.

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{2.8}$$

The manner in which the wavelet is dilated and translated depends on the type of wavelet transform being undertaken. Kostopouolos & Loutas (2012) states the most common wavelet transforms used in condition monitoring are the continuous wavelet transform (CWT), the discrete wavelet transform (DWT) and the wavelet packet transform (WPT).

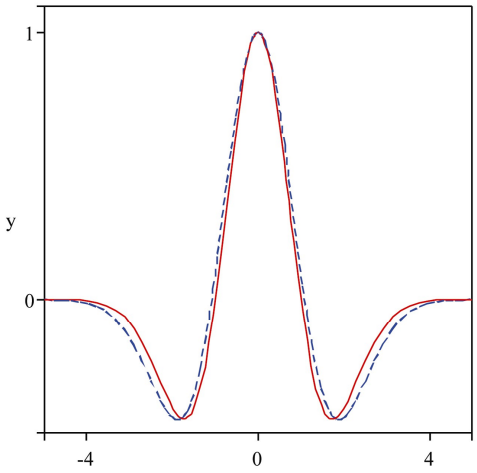

Figure 2.4: The general shape of the Mexican Hat wavelet.

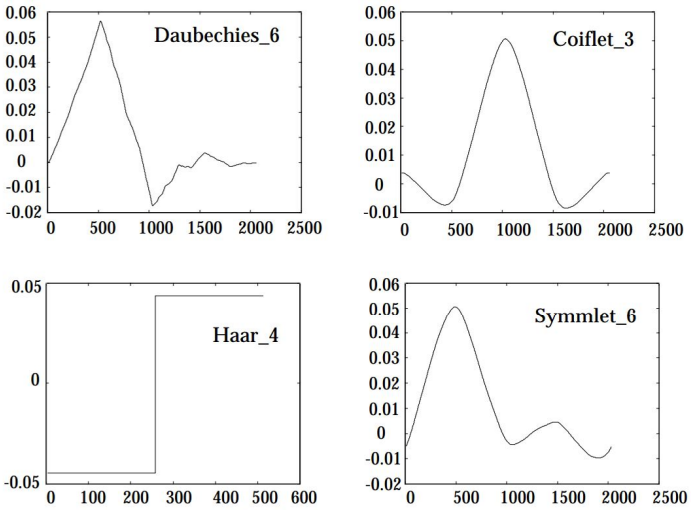

Figure 2.5: Examples of the Daubechies, Coiflet, Haar and Symmlet wavelet function shapes.

If the time series signal is continuous then the dilation and translation parameters may also be chosen to be continuous and this is referred to as the CWT. This is expanded upon in the following equations taken from (Goyal, Kovačević & Vertterli 2013). Consider a wavelet $\psi(\mathrm{t}) \in \mathcal{L}^2\ (\mathbb{R})$ centred on t=0, where $\mathcal{L}^2\ (\mathbb{R})$ represents a finite energy function. Now consider all possible translations and dilations,

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{2.9}$$

where $u \in +\mathbb{R}$ and $s \in \mathbb{R}$

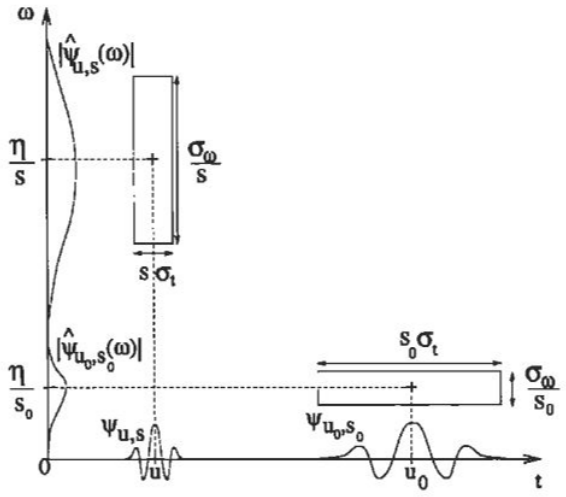

Figure 2.6: Two wavelet atoms with differing dilation and scaling. The X axis is time and the Y axis is frequency (Mallat 1998).

The wavelet is centred on u and scaled by factor s. For a given time signal $x(t)$, the CWT is given as,

$$X(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} \psi\left(\frac{t-u}{s}\right) x(t)dt \tag{2.10}$$

Where the time series signal is made up of discrete samples, such as a digital recording of an analogue signal, the DWT is used. The DWT is a discretized form of the CWT and is performed using a discrete matrix of dilation and translation parameters (Mohanty, Prabhakar & Sekhar 2002).

Mohanty et al. (2002) reports the most common form of the DWT is dyadic and is defined as,

$$DWT(j,k) = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{\infty} \Psi^*\left(\frac{t-2^j k}{2^j}\right) x(t)dt \tag{2.11}$$

where dilation and translation parameters s and u are replaced by 2^j and $2^j k$ making them integer values.

The textbook by Mallat (1998) presents a more efficient method to implement the DWT which takes lass computation time. Mallat discretized the wavelet function according to

the equation,

$$\psi_{j,k} = 2^{\frac{j}{2}}\psi(2^j t - k) \tag{2.12}$$

and created a discrete scaling function (dilation function),

$$\phi_{j,k} = 2^{\frac{j}{2}}\phi(2^j t - k) \tag{2.13}$$

where

$$f(x) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & otherwise \end{cases} \tag{2.14}$$

The discrete wavelet function is used to form a wavelet filter, a high pass filter g(n). The makeup of the filter depends on the type of mother wavelet selected. The discrete scaling function is use to form the scaling filter, h(n). Again, the details of the filter depend on the mother wavelet selected.

In Mallats more efficient DWT method the time signal is convolved with low pass and high pass filters to produce two vectors cA and cD. The elements of vector cA are called the *approximation coefficients* and the elements of vector cD are called the *detail coefficients*. The detail coefficients provide information about the upper half of the frequency band of the time signal. The approximation coefficients provide information about the lower half of the frequency band of the time signal.

Down sampling of each vector through omitting each odd indexed element reduces the number of elements to the same number as the sampled time signal (Antoniadis & Nikolaou 2002). The elements in the approximation vector can then be passed through the low and high pass filters again for a second level of signal decomposition, which divides the approximations into a subset of detail and approximation coefficients. This procedure is repeated until the desired level of decomposition is reached. This is shown in Figure 2.7.

The WPT again makes use of the wavelet and scaling functions, and also the high and

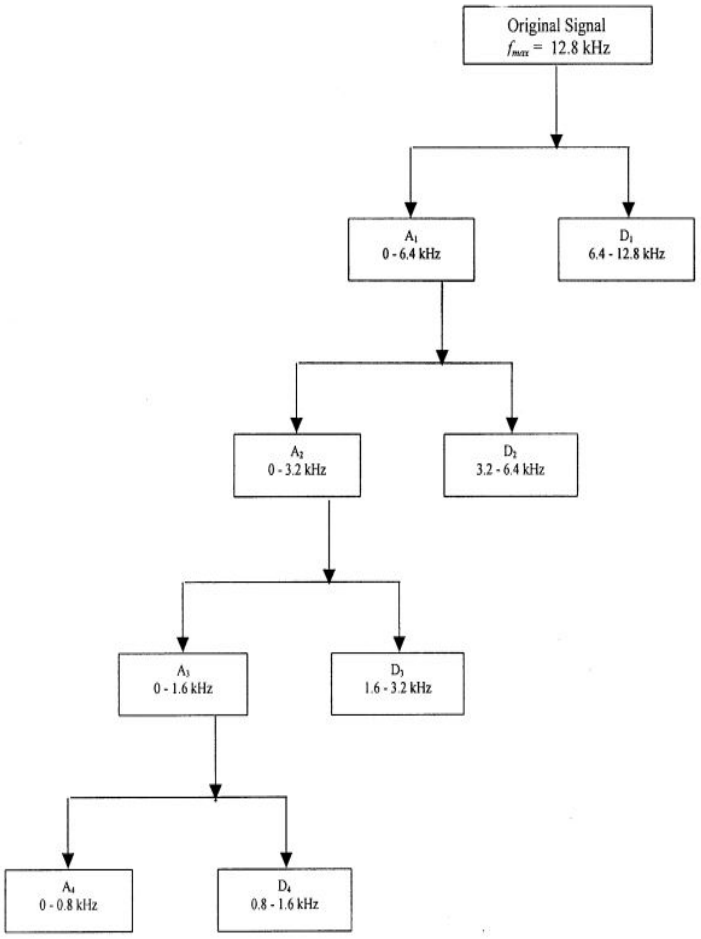

Figure 2.7: Successive decomposition of the original signal into approximation and detail coefficients produces this tree like structure. At the first level of decomposition the detail coefficients contain information about the original signal in the 6.4-12.8 kHz frequency band. The approximation coefficients contain information about the 0-6.4 kHz frequency band and are divided once again at the second level of decomposition. (Mohanty et al. 2002).

low pass filters unique to the mother wavelet type. The scaling function is defined as,

$$W_0(t) = \phi(t) \tag{2.15}$$

and the wavelet function is defined as,

$$W_1(t) = \psi(t) \tag{2.16}$$

The function *W(t)* for *m=1, 2, 3...*can be obtained as,

$$W_{2m}(t) = 2 \sum_{n=0}^{2N-1} h(n) W_m(2t - n) \tag{2.17}$$

$$W_{2m+1}(t) = 2 \sum_{n=0}^{2N-1} g(n) W_m(2t - n) \tag{2.18}$$

where j is the scaling parameter and n is the localisation parameter (Antoniadis & Nikolaou 2002).

The WPT decomposes the time signal in the same manner as the DWT with the exception that both details and approximation vectors are further split into detail and approximations. Figure 2.8 shows how this process creates a wavelet packet tree.

Each node of the tree is a vector of wavelet packet coefficients and is indexed with integers (j,k), where j is the level of decomposition and k is the order of node position in the level.

Works by Kalista & Liska (2015), Chandra & Sekhar (2016) and Gade & Gram-Hansen (1996) all used CWT successfully for detecting fault signals in bearing vibrations however, Kalista & Liska (2015) do make note that the CWT does take high computational time. Mohanty et al. (2002) and Kasashima, Mori, Ueno & Yoshioka (1996) used the DWT to detect race faults in bearings and predict spalling. Al-Badour et al. (2010) used the WPT to analyse the transient of bearing faults during electric motor start-up and coast-down. Antoniadis & Nikolaou (2002) used the WPT to extract a variety of bearing fault signatures from a signal.

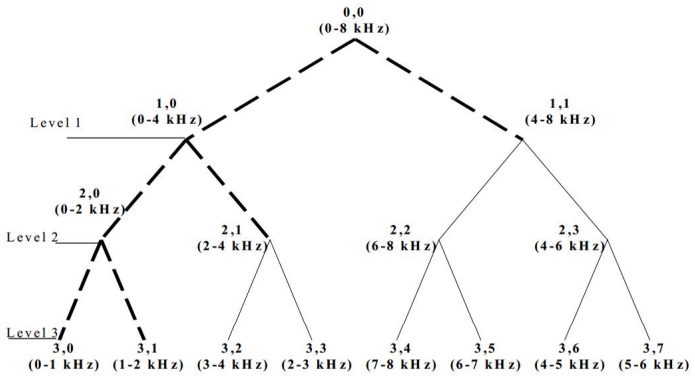

Fig. 3. An example of a three-level wavelet packet decomposition tree.

Figure 2.8: The WPT tree is similar to the DWT tree with the exception that both approximation and detail coefficients are further divided at each level of decomposition (Antoniadis & Nikolaou 2002).

2.5 Wavelet De-noising

An alternative use of wavelets in the condition monitoring context is to de-noise a signal prior to processing. In the industrial environment various types of mechanical and electrical noise are prone to corrupting the underlying signal, increasing the difficulty of detecting failing components. Wavelet de-noising works differently to standard signal filtering methods. Where traditional signal filtering the methods would remove a band or bands of frequencies from the signal, using pass band filtering, the goal of wavelet de-noising is to obtain an estimate of the original signal through the removal of selected frequency components (Cai & Harrington 1998).

The literature shows the underlying wavelet de-noising method is essentially a three step process (Cai & Harrington 1998, Zhang & Liu 2010, Luo & Zhang 2012).

1. The first step is to convert the signal to be analysed from the time domain into the wavelet domain. This process is explained in Chapter 2 Section 2.4.

2. Within the wavelet domain certain wavelet coefficients are zero-filled or reduced in magnitude based on some criterion. The criterion, referred to as thresholding, shall be further defined below but main function of this step is the artificial adjustment of time frequency components which removes the noise.

3. The final step is to inversely transform the adjusted wavelet coefficients to re-create the time-domain signal.

The literature reveals a wide range of methods and techniques that define how the second step in the above process works. Essentially these methods are centered on two questions; which coefficients should be selected for adjustment; and how much should they be adjusted by.

The concept of coefficient selection is referred to as thresholding. The threshold is simply a scalar limit to which each of the wavelet coefficients are compared against (Ergen 2012). The aim of the threshold is to remove components from the underlying signal in order to create the best possible estimation of the original signal (Luo & Zhang 2012) so in this context it is noted that wavelet thresholding is only providing an estimation of the noise content. According to the literature (Luo & Zhang 2012, Ergen 2012, Cai & Harrington 1998, Lin, Zuo & Fyfe 2004, Abbasion, Rafsanjani, Farshidianfar & Irani 2007, Lin 2001, Staszewski 1998) the three most commonly used methods for thresholding are:

- Universal.
- Minimax.
- Stein's Unbiased Risk Estimation (SURE).

Many variations of these three fundamental threshold techniques have been developed.

The Universal threshold was developed by Donoho & Johnstone (1994). It requires knowledge of the standard deviation of noise which is not normally known (Cai & Harrington 1998) so in practice a rough measure of the noise is used resulting I mixed results (Cai & Harrington 1998). The Universal threshold is defined as,

$$t_{universal} = \sigma\sqrt{2ln(N)} \tag{2.19}$$

where N is the length of the array and σ is the standard deviation of the noise. σ can be

estimated as s where,

$$s = \frac{median(|x_i|)}{0.6745} \tag{2.20}$$

The minimax threshold was initially proposed by Donoho & Johnstone (1998) as an optimal threshold derived through minimising a constant term within an upper limit of risk associated with estimating a function. The terminology 'risk' can be thought of as the difference between the actual function and the estimation of the function. The minimax threshold uses available data and attempts to account for contaminating noise by optimising a threshold that realises the minimum risk (Ergen 2012).

The optimal threshold is defined as,

$$t_{minimax} = \sigma\lambda_n \tag{2.21}$$

where λ_n is defined as the value of λ that satisfies,

$$\lambda_n = inf_\lambda sup_x \left\{ \frac{R_\lambda(d)}{n^-1 + R_{oracle}(d)} \right\} \tag{2.22}$$

where $R_\lambda(d)$ is the mean square error,

$$R_\lambda(d) = E(\tilde{x} - x)^2 \tag{2.23}$$

and $\tilde{x}$ is the estimation of the function.

R_{oracle}(d) is used to account for the risk associated with modifying a wavelet coefficient (Ergen 2012). The work by Donoho & Johnstone (1998) originally proposed two options for the oracle, a diagonal linear projection (DLP) and a diagonal linear shrinker (DLS). The ideal risks given by these oracles are,

$$R^{DLP}_{oracle}(d) = min(d^2, 1) \tag{2.24}$$

$$R_{oracle}^{DLS}(d) = \frac{d^2}{d^2+1} \tag{2.25}$$

SURE threshold was initially developed by Stein (1981) to reduce the risk of incorrect signal estimation by lowering the threshold level (Luo & Zhang 2012). SURE thresholding is used to derive the unbiased estimation of the signal noise and is defined as,

$$SURE(t,x) = N - 2M_{(i:|x_i|\leq t)} + \sum_{i=1}^{N}(|x_i| \wedge t)^2 \tag{2.26}$$

where t is the candidate threshold, x_i is the wavelet coefficient, N is the data size and M is the number of data points smaller in magnitude than threshold t (Cai & Harrington 1998).

The *unbiased* nature of the threshold results from the argument that for a given signal consisting of a large number of samples and kind of statistical regularity may set in (Cai & Harrington 1998).

Once a threshold level is selected the question then shifts towards how best to apply the threshold. A number of options are presented by the literature. Hard thresholding is the simplest method where all wavelet coefficients below the threshold are deemed to be non-significant, in terms of the underlying signal trying to be estimated, and are therefore set to zero (Ergen 2012, Luo & Zhang 2012, Cai & Harrington 1998). All wavelet coefficients above the threshold remain at their original wavelet transformed magnitude. Hard thresholding is defined as,

$$x_i^* = \begin{cases} 0 & \text{if } |x_i| \leq \text{t} \\ x_i & \text{if } |x_i| > \text{t} \end{cases} \tag{2.27}$$

where x_i^* is the wavelet coefficient before de-noising and x_i is the wavelet coefficient after de-noising.

Hard thresholding can introduce discontinuities within the de-noised data but can produce smaller RMS errors than other methods (Cai & Harrington 1998). The issue of discontinuous signal is thought to be insignificant when dealing with typically random signals such as vibration signals.

Soft thresholding treats wavelet coefficients below the threshold in the same manner as hard thresholding. However, for wavelet coefficients that are above the threshold value the soft thresholding method adds or subtracts the threshold value to or from the wavelet coefficient value to reduce the magnitude of the coefficient (Donoho 1995).

$$x_i^* = \begin{cases} 0 & \text{if } |x_i| \leq \text{t} \\ sign(x_i)(|x_i| - t) & \text{if } |x_i| > \text{t} \end{cases} \tag{2.28}$$

Soft thresholding retains signal continuity however it can generate larger RMS errors than hard thresholding. Soft thresholding can over smooth abrupt changes and sharp peaks which may be appropriate for slower moving signals however may not suite fast moving data such as vibration signals (Cai & Harrington 1998).

Chapter 3

Implementation in MATLAB^{TM}

3.1 Chapter Overview

This short chapter should be read in conjunction with Appendix B.1 and serves to explain some of the MATLAB^{TM} code used to implement theory presented in the literature review. The chapter will reserve the explanation of the program development for Chapter 5 Methodology. When writing the main program in this dissertation I also wrote smaller functions which the main program calls during operation. These functions are discussed here. The chapter concludes with some explanations of MATLAB^{TM} built in functions used including wavelet de-noising parameters.

3.2 Frequency Extraction Methods

Three frequency extraction methods were identified in the literature review for use in this study. The methods are standard signal processing techniques and based on Fourier analysis, envelope analysis and cepstrum analysis.

Fourier analysis analyses the time domain signal for frequency content. This achieved in MATLAB^{TM} using the absolute function *abs()* which returns the absolute value in an array, and the fast Fourier transform function *fft()* which efficiently computes the discrete Fourier transform.

Envelope analysis works in a very similar manner to Fourier analysis and the structure of the code is identical with the exception that envelope analysis processes a signal called the analytic signal. The analytic signal can be thought of as two signals, one on the real axis and one on the imaginary axis. The signal on the real axis is the original time domain signal. The signal on the imaginary axis the original signal phase shifted -90°. In MATLAB^{TM} the analytic signal is created using the built in Hilbert transform function *hilbert()*. The analytic signal is then analysed using Fourier analysis.

Cepstrum analysis is created in MATLAB^{TM} using the functions *fft()*, *abs()*, *log()* and the inverse fast Fourier transform *ifft()*.

The full MATLAB^{TM} program is given in Appendix B.1 and shows how these functions are combined to perform these frequency extraction methods.

3.3 Wavelet De-noising in MATLAB^{TM}

The MATLAB^{TM} Wavelet Toolbox contains the wavelet de-noising function *wden()* which performs automatic 1-D de-noising of a one-dimensional signal using wavelets. *wden()* requires six inputs one of which is the signal being de-noised. The remaining five inputs are the wavelet de-noising parameters that are explored in this dissertation. MATLAB^{TM} has set parameters when using *wden()*. Information explaining these parameters has been extracted from the MATLAB^{TM} Wavelet Toolbox Users Guide (Misiti, Misiti, Oppenheim & Poggi 2016).

The first parameter is the type of threshold. The MATLAB^{TM} options here are:

- 'rigrsure' which uses Stein's Unbiased Risk Estimate.
- 'sqtwolog' which is the universal threshold method.
- 'heursure' which stands for heuristic SURE and is a combination of the above two thresholds.
- 'minimaxi' which uses the minimax principle.

The second parameter is the setting of soft or hard thresholding. The third parameter

sets the noise scale estimation. Three options are available. '*One*' essentially treats the input signal as though it complies to the basic noise model of,

$$s(n) = f(n) + e(n) \tag{3.1}$$

where *f(n)* is the desired signal and *e(n)* is the Gaussian random variables that form noise.

'*Sln*' uses a single estimation of noise based on the wavelet coefficients from the 1st level of wavelet decomposition. '*Mln*' uses multiple estimations of noise based on each level of wavelet decomposition. This is typically used in circumstances where the noise is suspected to non-white noise. The fourth parameter is the level of wavelet decomposition the wavelet is to process down to and the fifth parameter is the type of wavelet to be used in the process. In this dissertation the family of wavelets used is the Daubechies wavelets and the parameters used here range from DB1 to DB19.

3.4 Other Functions Written

In support of the main program there were three additional MATLAB^{TM} functions written. One function, called `freq_limit.m`, adjusts the frequency bandwidth from the full frequency range down to the desired bandwidth. The desired lower and upper bounds of the frequency bandwidth are passed to the function as well as the frequency spectrum requiring adjusting. The function returns the adjusted frequency spectrum.

The other two functions are very similar in nature. They are titled `freq_energy_calc.m` and `cep_energy_calc.m`. They both assist in calculating the signal energy from the frequency and quefrency domains. The inputs to the functions are the frequency (or quefrency) spectrum, the lower frequency limit and the upper frequency limit. The function searches for the peak frequency component within the lower and upper limits. It then sums the frequency energy of a smaller bandwidth surrounding the peak frequency component and returns this value as the energy of one of the characteristic fault frequencies.

The MATLAB^{TM} code for these functions are provided in Appendix B.

Chapter 4

Signals

4.1 Chapter Overview

This chapter describes the vibration signals used in this body of work, including both simulated signals and real signals. It provides simulated signal details, real vibration signal details acquired through measurement, bearing technical details, and testbed and arrangements.

To adequately understand the behaviour of the characteristic fault frequencies (CFF) methods devised, and to analyse and debug problems in the process, I required an artificial signal that simulated the behaviour of a vibration signal. The use of a simulation signal provided complete knowledge of the frequency content and periodicities present. A simulation signal also provided the opportunity to add varying levels of Gaussian noise to the clean signal in order to assess the CFF response under varying signal to noise ratio (SNR) levels. The derivation and limitation of the simulation signals used is described below.

Real vibration signal data was acquired from a number of websites whom make these signals publicly available. The real vibration data is categorised into Short Time signals and Long Time signals. Short time signals are vibration signals acquired under controlled laboratory conditions using an experimental testbed arrangement. A typical testbed arrangement involves seeding a bearing with a mechanical fault to artificially produce one of the CFFs. Short time signals are typically only a few seconds in duration and

provide a brief insight into the vibration signal produced by the bearing's operational wear at that moment in the bearing life cycle. The level of mechanical damage does not vary during the course of the short data recording.

The long time signals used in this work consist of vibration data acquired over a period of 32 days. This dataset recorded the vibration signals of a bearing over a longer duration and observed the change in vibration profile as the mechanical damage in the bearing increased in magnitude. Long time signals provide vibration data at both early and advanced stages of bearing failure.

A total of 10 GB of vibration data was accessed at the beginning of this project although not all data files were used. These datasets were acquired from many different experimental and real world acquisition setups. For this reason it was decided that this project did not require the construction of a bearing testbed for acquiring additional vibration signals.

4.2 Simulation Signals

During the course of the literature review (Chapter 2) a number methods were identified for creating simulation signals. Kiral & Karagülle (2003) used Finite Element Analysis (FEA) to model the vibration frequency response when a bearing element rolls across a mechanical defect. Works by McInerny & Dai (2003) and Lin & Qui (2000) explains the generation of a synthetic signal via simple mathematics.

The product of the works by (Kiral & Karagülle 2003, McInerny & Dai 2003, Lin & Qui 2000) were signals that shared similar characteristics to that of a faulted bearing vibration signature. On this basis a simulation signal was created to mimic a vibration signal. The simulation signal consisted of the following parameters:

- A series of decaying cosine pulses, 6 ms in duration. During each pulse the cosine function cycles approximately 10 times as it decays to zero.
- The time period between pulses is 14 ms which means the pulse starts every 20 ms.
- The total length of the pulse series is 10 seconds (the length of the first iteration of the simulation signal was only 4000 samples long equating to 1/5 of a second).

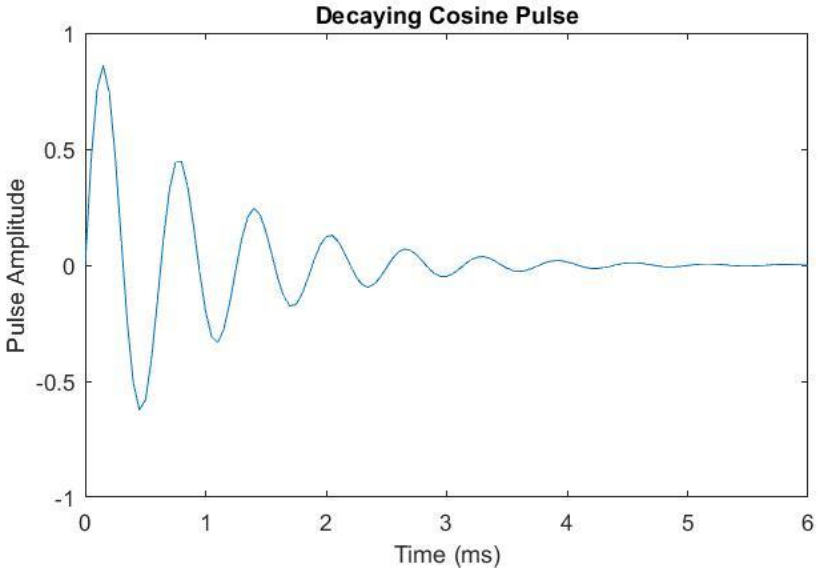

Figure 4.1: Decaying cosine pulse 6 milliseconds in duration.

- The sample frequency was set to 20,000 Hz.

The pulse is shown in Figure 4.1 and a portion of the simulation signal is shown in Figure 4.2.

Since a new pulse occurs every 20 ms it is clear that the pulse frequency is 50 Hz. Further, the cosine function cycles approximately 10 times in 6 ms therefore significant frequency content is expected to be in the vicinity of 1666 Hz, as calculated by,

$$Frequency = \frac{1}{6ms}10 = 1666.67Hz \tag{4.1}$$

However in practice, when calculating the frequency spectrum of the simulation signal the abrupt nature of the start of the pulse must be considered and results in significant spectral leakage and smearing of the dominant 1666 Hz component. Figure 4.3 shows the full frequency spectrum of the noise free simulation signal. As the sample frequency for this signal is 20,000 Hz, the full frequency spectrum ranges from 0 Hz to 10 kHz.

The simulation signal described above formed a baseline signal with which to analyse the

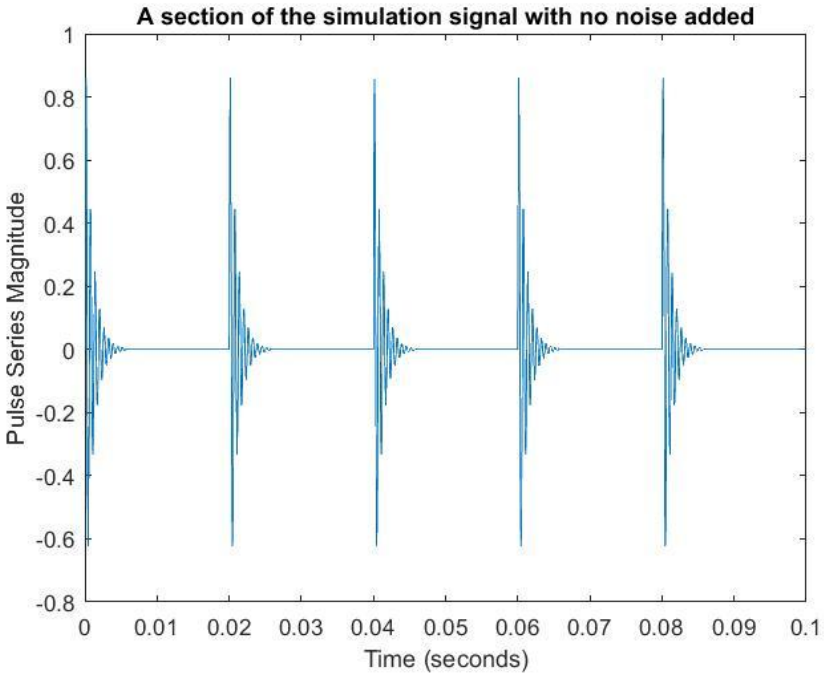

Figure 4.2: A portion of the simulation signal.

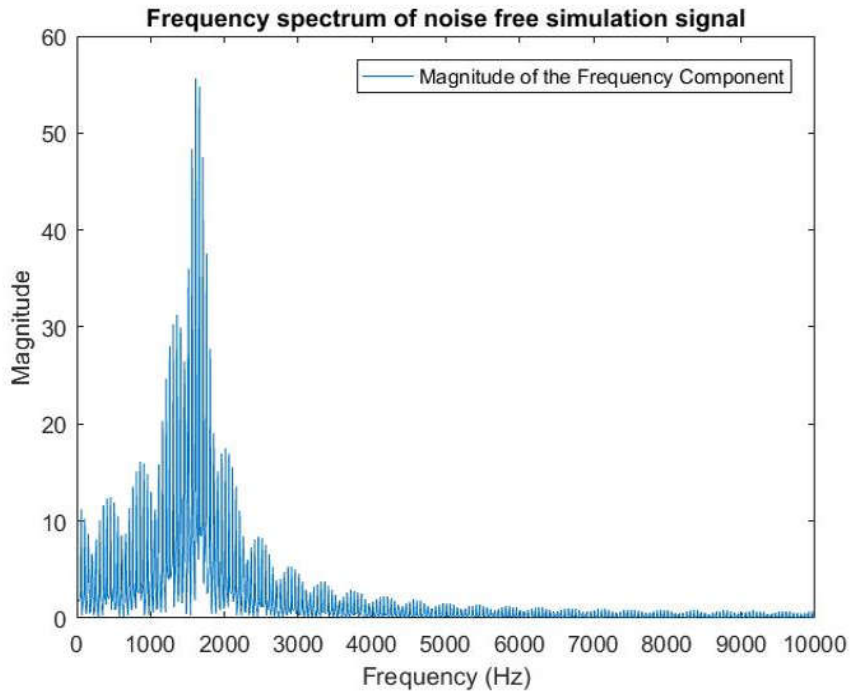

Figure 4.3: The full frequency spectrum of the simulation signal.

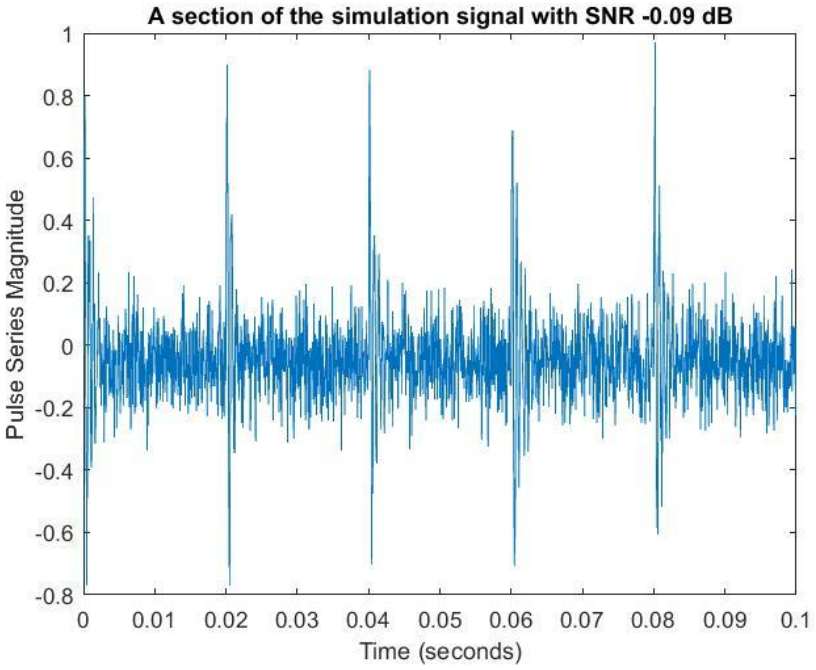

Figure 4.4: A portion of the simulation signal with SNR -0.09 dB.

performance of the 3 CFF methods. The analysis of the 3 CFF methods also included assessing their performance under varying levels of SNR. Using the basic simulation signal as the starting point I created a range of 'noisy' signals through vector addition of a noise vector containing Gaussian distributed random numbers centered around zero. To vary the SNR of the combined signal the magnitude of the noise vector was increased in discrete steps ranging from 0.1 to 0.8.

Figure 4.4 shows the simulation signal with SNR of 0.09 dB. This signal had Gaussian noise of magnitude 0.1 added to it.

This method permitted very accurate calculation of the noisy simulation signal's SNR. As Lyons (2011) indicates in his text book, the method for calculating SNR is,

$$SNR = \frac{SignalPower}{NoisePower} = \frac{\sum_{n=0}^{N-1}[x_s(n)]^2}{\sum_{n=0}^{N-1}[x_n(n)]^2} \tag{4.2}$$

where x_s are the real valued signal samples and x_n are the real valued noise samples.

Calculating the SNR of the noisy simulation signals is made easy since there is complete knowledge of the underlying simulation signal and the magnitude of the noise being added.

4.3 Short Time Signals

Short time signals are signals acquired from a bearing testbed arrangement under laboratory conditions. The vibration signal is short in duration often lasting only a few seconds therefore it does not offer any differing vibration profile as the severity of the mechanical fault increases. The mechanical faults are often artificially seeded in the bearing and produce strong frequency content relating to the CFF.

4.3.1 Case Western Reserve Bearing Data Centre

Case Western Reserve University is located in Cleveland Ohio. 161 bearing vibration files were accessed from the university online Bearing Data Centre on 17 October 2015. Acknowledgement is made for the measurements used in this work provided through Case Western Reserve Bearing Data Centre database (Case Western Reserve University 2015).

The details of the bearings used in these files are summarised below:

Drive end bearing:	6205-2RS JEM SKF
Inside diameter:	0.9843”
Outside diameter:	2.0472”
Thickness:	0.5906”
Ball diameter:	0.3126”
Pitch diamter:	1.537”

Fan end bearing:	6203-2RS JEM SKF
Inside diameter:	0.6693”
Outside diameter:	1.5748”
Thickness:	0.4724”
Ball diameter:	0.2656”
Pitch diamter:	1.122”

Using this information the characteristic fault frequencies were calculated to be:

Characteristic Fault Frequency	**Multiple of shaft speed (in Hz)**
Drive end bearing	
Inner Race Defect Frequency:	5.4152
Outer Race Defect Frequency:	3.5848
Ball Spinning Frequency:	4.7135
Cage Frequency:	0.3982
Fan end bearing	
Inner Race Defect Frequency:	4.9469
Outer Race Defect Frequency:	3.0530
Ball Spinning Frequency:	3.9874
Cage Frequency:	0.3817

The Case Western Reserve dataset offers a wide range of vibration signals where the type of fault, the level of damage and the bearing load are all varied. The varied bearing load also changed the shaft speed. The dataset consists of rolling element faults, inner race faults and outer race faults. Vibration data was recorded for each fault with four different loads applied which in turn varied the shaft speed. Mechanical faults were artificially seeded in the bearing through electro-discharge machining one of bearing components. The depth of the damage remained constant but the diameter of the damage was varied from 0.007" to 0.028" to produce different vibration responses.

Case Western Reserve also provide details of the bearing testbed arrangement. Figure 4.5 shows a 2 hp motor driving a torque transducer which in turn drives a dynamometer. The bearings being monitored are supporting the motor shaft. The data files used two data acquisition sample frequencies, either 12,000 Hz or 48,000 Hz.

4.3.2 data-acoustics.com

data-acoustics.com is a project to provide free access to a curated set of vibration measurement data. Multiple entities have made contributions to the website. Acknowledgement is made for the measurements used in this work provided through data-acoustics.com database (data-acoustics 2015).

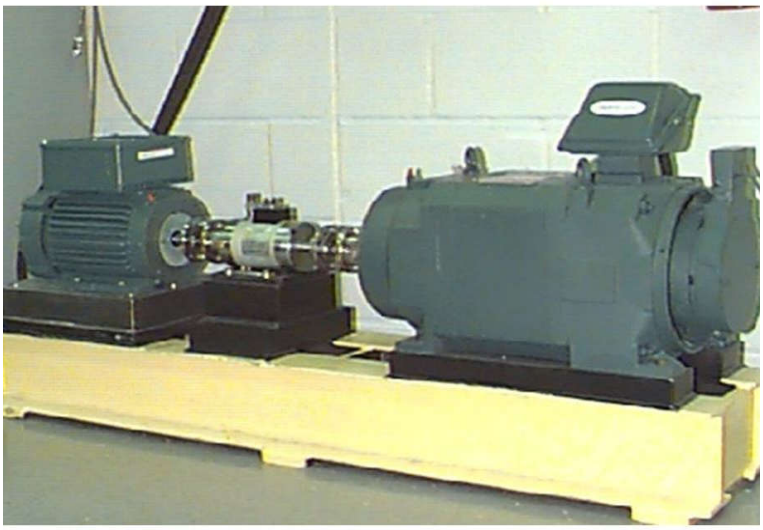

Figure 4.5: The Case Western Reserve bearing testbed. (Case Western Reserve University 2015)

Inner and Outer Race Bearing Fault Vibration Measurements

Forbes (2012) of Curtin University, Western Australia published two vibration data measurements on the data-acoustics.com website.

The data provided by Forbes (2012) was acquired using a SpectraQuest Machinery Fault Simulator. The bearing and data acquisition parameters used by Forbes (2012) during his work are summarised below:

Bearing Number:	MB ER-16K
Number of balls:	9
Ball Diameter:	7.94 mm
Pitch Diameter:	38.5 mm
Data Length:	10 seconds
Sample Rate:	51,200 Hz
Shaft Speed:	29 Hz

Using this information the characteristic fault frequencies were calculated to be:

Characteristic Fault Frequency	Frequency	Quefrency
Inner Race Defect Frequency:	157.41 Hz	6.35 ms
Outer Race Defect Frequency:	103.59 Hz	9.65 ms
Ball Spinning Frequency:	67.28 Hz	14.86 ms
Cage Frequency:	11.51 Hz	86.88 ms

Two data sets were produced, one with a seeded fault in the inner race and the other with seeded fault in the outer race. Details of the magnitude of the mechanical damage seeded in the inner and outer bearing races were not published.

4.3.3 Machinery Failure Prevention Technology

The Machinery Failure Prevention Technology (MFPT) group is a division of the not-for-profit corporation Vibration Institute. MFPT is an interdisciplinary organisation geared towards practical application of machinery health management systems. Acknowledgement is made for the measurements used in this work provided through mfpt.org database (Machinery Failure Prevention Technology 2015).

Bechhoefer (2012) supplied to the MFPT website 20 different testbed vibration data files grouped in four different categories.

The same bearing type was used for each data file:

Bearing Type:	NICE
Number of balls:	8
Ball Diameter:	5.97 mm
Pitch Diameter:	31.62 mm
Contact Angle:	0
Shaft Speed:	25 Hz

Using this information the characteristic fault frequencies were calculated to be:

Characteristic Fault Frequency	Frequency	Quefrency
Inner Race Defect Frequency:	118.88 Hz	8.41 ms
Outer Race Defect Frequency:	81.13 Hz	12.32 ms
Ball Spinning Frequency:	63.91 Hz	15.65 ms
Cage Frequency:	14.84 Hz	67.39 ms

The details of the four categories Bechhoefer (2012) grouped the data files into are summarised below:

Baseline case - no fault:	Sample rate:	97,656 Hz
	Record length:	6 seconds
	Load:	270 lbs for each of the 3 files
Outer race faults:	Sample rate:	97,656 Hz
	Record length:	6 seconds
	Load:	270 lbs for each of the 3 files
Outer race faults - variable load:	Sample rate:	48,828 Hz
	Record length:	3 seconds
	Load:	Varied for each of the 7 files
Inner race faults - variable load:	Sample rate:	48,828 Hz
	Record length:	3 seconds
	Load:	Varied for each of the 7 files

The inner and outer race faults in the data files were artificially seeded with mechanical damage. No technical details were provided on the level of damage or the means of inflicting the damage however Figure 4.6 and Figure 4.7 provide a visual indication of the type of damage to the inner and outer bearing races.

Figure 4.6: Artificial mechanical damage applied to the bearing inner race (Bechhoefer 2012).

Figure 4.7: Artificial mechanical damage applied to the bearing outer race (Bechhoefer 2012).

4.4 Long Time Signals

Long time signals are a sequential set of vibration data files, acquired over a period of days or weeks, and track the bearing vibration profile as the mechanical fault increases in severity. Each file is still only a few seconds in duration but many files are acquired in sequence as bearing wear increases. The bearings used in these files can start out as brand new or can be seeded with a fault.

4.4.1 Intelligent Maintenance Systems

The dataset was recorded and published by the Center for Intelligent Maintenance Systems (Center for Intelligent Maintenance Systems 2015), University of Cincinnati with the assistance of Rexnord Corporation. However the data was downloaded from the National Aeronautics and Space Administration (NASA) Prognostics Data Repository (National Aeronautics and Space Administration 2015).

The same bearing type was used for each data file:

Bearing Type:	Rexnord ZA-2115 double row bearing
Number of balls:	16
Ball Diameter:	0.331”
Pitch Diameter:	2.815”
Contact Angle:	15.17 °
Shaft Speed:	33.33 Hz

Using this information the characteristic fault frequencies were calculated to be:

Characteristic Fault Frequency	**Frequency**	**Quefrency**
Inner Race Defect Frequency:	293.96 Hz	3.40 ms
Outer Race Defect Frequency:	236.38 Hz	4.23 ms
Ball Spinning Frequency:	139.90 Hz	7.15 ms
Cage Frequency:	14.77 Hz	67.70 ms

Details were also provided for the bearing testbed arrangement used for these files. Four bearings were installed on a shaft and the shaft was radially loaded with 6000 lbs. Each

bearing housing had one PCB 353B33 high sensitivity quartz ICP accelerometer installed. The signals were acquired using a sample frequency of 20,000 Hz Figure 4.8 shows the testbed arrangement used.

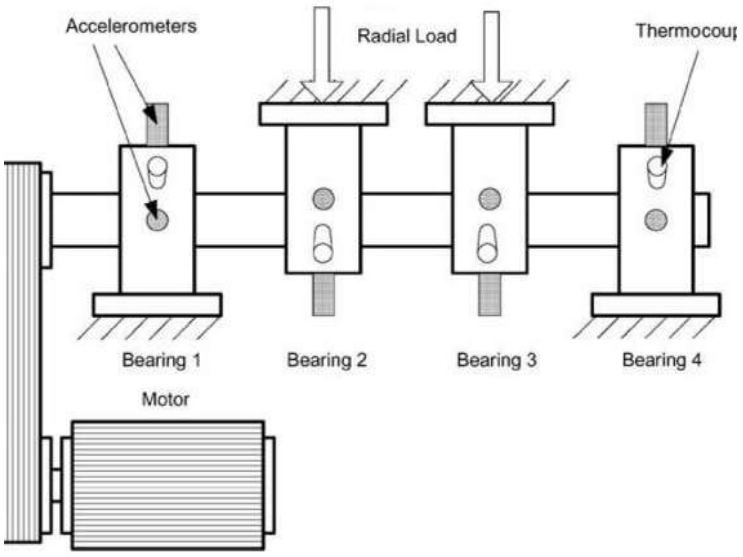

Figure 4.8: The IMS bearing testbed arrangment (Qiu et al. 2006).

Chapter 5

Methodology

5.1 Chapter Overview

This chapter begins with an explanation of the methods developed for quantifying signal improvement in order to assess the wavelet de-noising process. Three methods were tried and tested and ultimately a solution was chosen that could assess both simulated and real vibration signals. The chapter then discusses the development of the three methods for extracting the characteristic fault frequencies (CFF) focusing on the development stages progressing from analysing a single frame of data, then on to multiple frames of data and then on to real vibration data signals.

5.2 Quantifying Signal Improvement After De-noising

Assessing the effect wavelet de-noising has on the vibration signal corrupted by noise requires a method of measuring the signal before and after the de-noising process. Three methods were developed and trialled during the course of the project. The first method used knowledge of the underlying noise free signal and calculated the signal to noisre ratio (SNR) using traditional means. This method was used on simulation signals only. The second method analysed 1000 frames of data before and after the de-noising process. The number of times the correct peak frequency component was selected before and after de-noising formed the metric to assess the de-noising step. An increase in the number of correct frequency components indicated an improvement in the signal quality from

the de-noising process. The third method calculated the SNR from within the frequency domain which negated the requirement for knowledge of the underlying signal or noise level.

5.2.1 Traditional SNR

During the early stages of the project I was working exclusively with artificial signals designed to simulate real vibration signals. Chapter 4 provides an explanation of the construction of these signals. When constructing these signals I had control of the underlying noise free signal and also the magnitude of Gaussian noise added. With knowledge of both the underlying signal and the added noised I could calculate the SNR before de-noising using,

$$SNR = \frac{SignalPower}{NoisePower} = \frac{\sum_{n=0}^{N-1}[x_s(n)]^2}{\sum_{n=0}^{N-1}[x_n(n)]^2} \tag{5.1}$$

where x_s are the real valued signal samples and x_n are the real valued noise samples.

To make comparison between the signals before and after the de-noising process I needed the SNR of the de-noised signal. To calculate the SNR after the de-noising process I made the assumption the underlying signal was largely unaffected by the wavelet de-noising process. On this basis I could subtract the original noise free signal from the de-noised signal and the remaining signal components would comprise of the remaining noise vector after the de-noising stage. This concept is summarised in the following equations.

When constructing the simulation signal,

$$x_{Noisy}(n) = x_{Signal}(n) + x_{Noise}(n) \tag{5.2}$$

therefore to calculate the remaining noise after de-noising,

$$x_{De-noised}(n) - x_{Signal}(n) = x_{NoiseRemaining}(n) \tag{5.3}$$

The above described method was tested by wavelet de-noising the simulation signals and in some cases revealed very large improvements in SNR after the de-noising process. These

were suspiciously large improvements in the signal quality and investigation revealed the above described method became flawed when the noisy signal was de-noised using a wavelet with a decomposition level of four or greater. It was identified that the underlying signal was being attenuated during the de-noising process and almost completely removed both the noise and the signal. The assumption I had made that the underlying signal would remain unchanged was incorrect and, coupled with the almost complete removal of signal noise, produce very large increases in SNR which did not accurately represent the effect of the wavelet de-noising process.

5.2.2 Statistical Assessment

Statistical assessment was the second method developed for assessing the effect of the wavelet de-noising process. The method essentially relies on the statistical probability that the correct peak frequency component is selected when analysing a frame of data. For example, the noise free simulation signal has strong frequency content at 1611 Hz when using Fourier analysis, and strong quefrency content at 20.05 ms when using cepstrum analysis. These would be the correct frequency/quefrency components when analysing the simulation signal. An improvement in the signal quality from the wavelet de-noising process should then increase the probability of the correct peak frequency component being selected.

During the development of the three CFF extraction methods, Gaussian distributed noise vectors were added to the noise free simulation signal for the purpose of testing the response. When the noisy signals were processed it was observed that the peak frequency component (or quefrency component when using cepstrum analysis) would change from one data frame to the next. This occurred despite the same magnitude of Gaussian noise being added each time. I had knowledge of the underlying simulation signal therefore I knew which frequency component should be the largest in magnitude, however the added Gaussian noise was causing the peak frequency component to change with each new noise vector added. On the odd occasion the correct peak frequency component was being selected.

I then conducted an experiment where I added 1000 unique Gaussian noise vectors of the same magnitude to a clean noise free simulation signal and recorded the peak frequency component each time. Out of the 1000 peak frequency components I counted how many

of those were the correct frequency component derived from the clean simulation signal. I then repeated the process but increased the magnitude of the Gaussian noise vectors. The second set of 1000 frames produced a marked decrease in the number of correctly chosen peak frequency components. I then proceeded to do the same experiment for 10 different magnitudes of Gaussian added noise ranging from 0.05 up to 0.9. Figure 5.1 shows the quantity of correctly chosen peak frequency components in relation to the level of Gaussian noise.

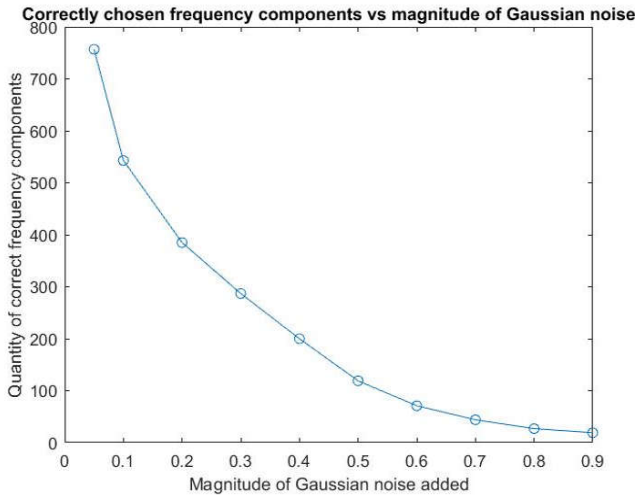

Figure 5.1: The quantity of correctly chosen peak frequency components decreases as the magnitude of Gaussian noise increases.

The conclusion I drew from this experiment was that the quantity of correctly chosen peak frequency components was inversely proportional to the magnitude of the Gaussian added noise. On the basis that the quantity of correctly chosen peak frequency components changed in relation to the level of signal noise I hypothesised that a noise removal process, such as wavelet de-noising, should increase the quantity of correct peak frequency components. I tested this theory by processing 1000 frames of simulation signal with Gaussian added noise. For each frame I recorded the peak frequency component and then wavelet de-noised the frame and recorded the new peak frequency component. The wavelet de-noising process showed an increase in the quantity of correctly chosen peak frequency components over the 1000 samples. Figures 5.2 and 5.3 are histograms demonstrating the shift in peak component distribution due to the wavelet de-noising process.

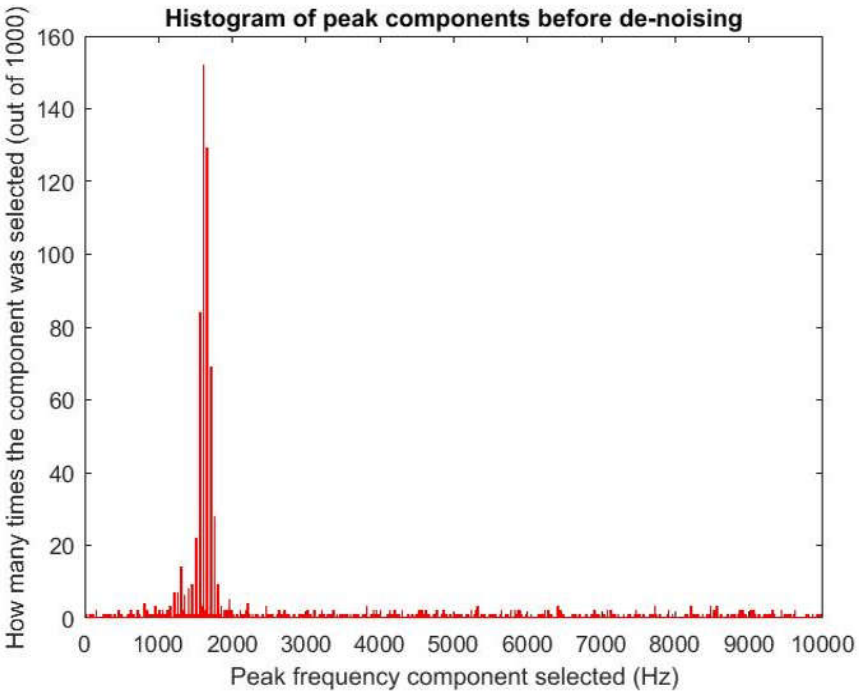

Figure 5.2: The histogram of the peak components with Gaussian noise magnitude of 0.6 - before de-noising.

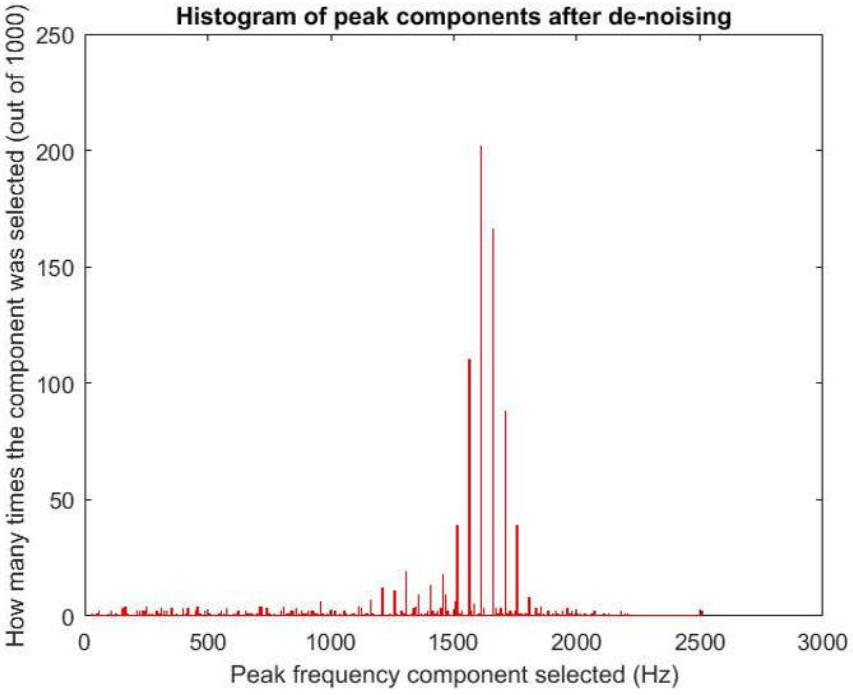

Figure 5.3: The histogram of the peak components with Gaussian noise magnitude of 0.6 - after de-noising.

Before using this method to assess all combinations of wavelet de-noising parameters against varying levels of Gaussian noisy simulation signals I decided to confirm whether the Statistical Assessment method would be appropriate for use with real vibration signals. Using the same method as described above I assessed 1000 frames of a signal acquired from a testbed arrangement recording the peak frequency component for each frame. I then de-noised each frame and recorded the new peak frequency. A problem was identified with this method in that the peak frequency component was the same for every noisy frame analysed and no statistical variation was occurring from one frame to the next. It was often the case that the peak frequency component was also the correct component. When the correct component was being selected at every frame there could be no measure of improvement resulting from the wavelet de-noising process. It is noted that in some cases the peak component was not the correct component however the peak component was so strong that wavelet de-noising had no effect on changing this to the correct component.

Analysis of this problem revealed that the strong frequency content in bearing signals acquired from a testbed arrangement was the result of the artificial seeding of mechanical damage within the bearing. These bearings were being disassembled and then artificially seeded with a fault on a single bearing component and then re-assembled. The magnitude of the mechanical damage was such that the characteristic fault frequency components relating to the damaged bearing were very strong and dominated the frequency/quefrency spectrum. For this reason the Statistical Assessment method could not be used to assess the effect of the wavelet de-noising process.

5.2.3 SNR in the Frequency Domain

The final method for assessing the effect of the wavelet de-noising process on the vibration signal, and the method used in this work relies on estimating the SNR from within the frequency domain. This idea was borrowed from Lyons (2011) where he describes a method that defines a noise threshold in the frequency domain and assumes all frequency components above the threshold are part of the signal, and all frequency components below are the noise. I slightly changed this idea and considered particular bands of frequencies as the desired signal, and all other frequencies classed as the noise.

Remembering that knowledge of the bearing geometry allows calculation of the bearing CFFs. Therefore, regardless of the state of mechanical damage of the bearing there will be

frequency content at these frequencies however the strength will be unknown. Summing and taking the power of small frequency bands surrounding each of the CFFs will produce the signal power. These are the frequencies we are trying to observe as shown in Figure 5.4. If we make the assumption that all other frequencies are not wanted then we can think of those frequencies as noise. The signal power is then divided by noise power to produce the SNR.

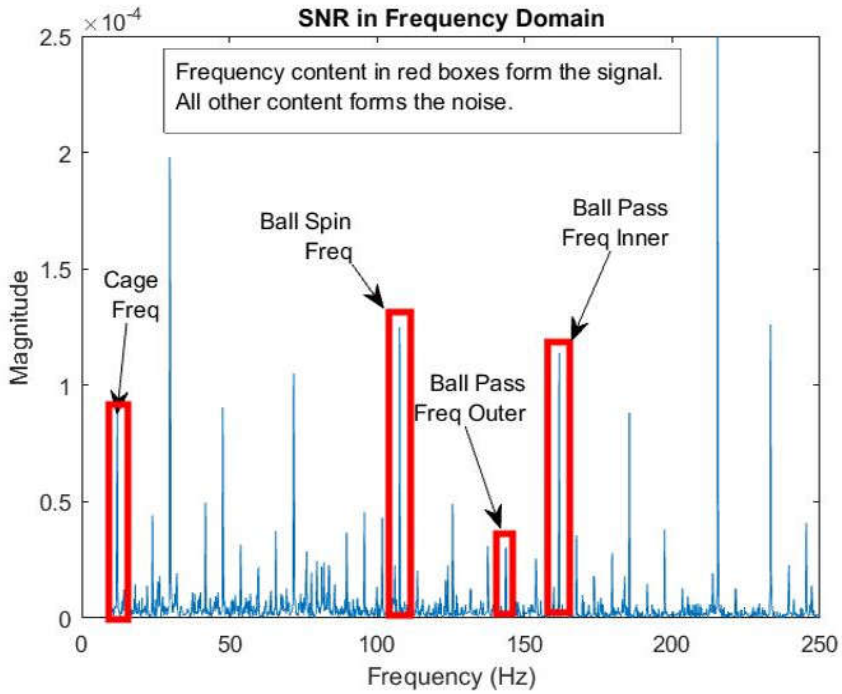

Figure 5.4: Summing the frequency content around the CFFs equates to the signal. All other frequency content is regarded as noise.

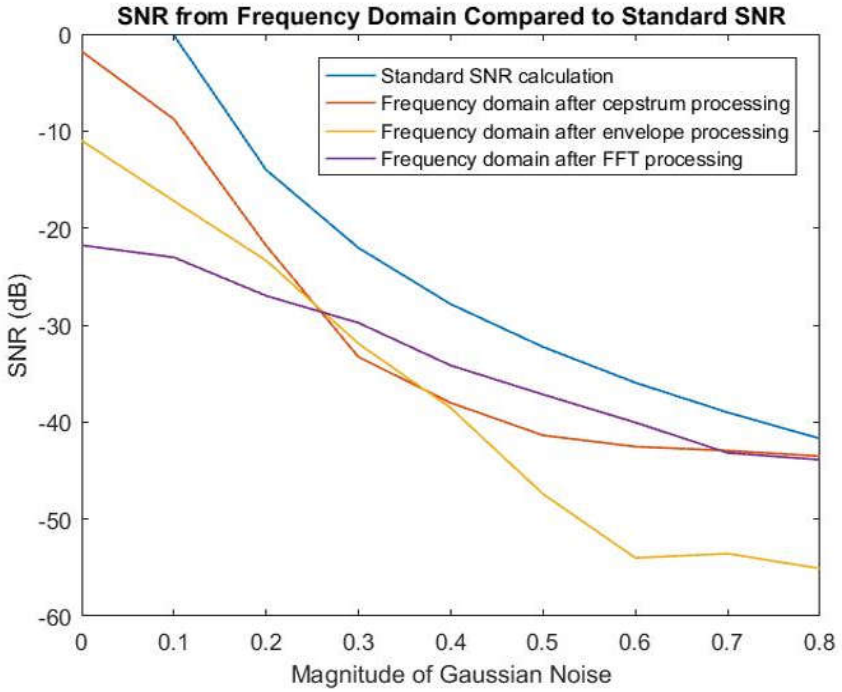

Figure 5.5: A comparison between the true SNR and the estimated SNRs taken from the frequency and quefrency domains for the simulation signal.

The advantage of this method is there is no requirement of knowledge of the underlying signal or noise power. However it is important to note that this method only provides an estimate of the true SNR and can therefore only be used for assessing the change in the strength of the desired frequency components with respect to the background noise. Consider the size of the bandwidth of the frequency or quefrency spectrum being observed. If the noise component of the SNR is made up of frequencies other than the CFFs then the size of the bandwidth being observed will directly influence the magnitude of the noise in the SNR equation. If the bandwidth is small and restricted around the 4 CFFs then the noise component of the SNR equation may be considered small. However if the bandwidth is large then a greater quantity of frequency components are contributing to the noise component of the SNR equation. For this reason this method of SNR calculation can only be used to estimate the SNR. Further, when comparing one SNR to another, the overall bandwidths and the CFF bandwidths must remain constant from one SNR estimation to the next in order for the comparison to be equal between the two.

Figure 5.5 shows the true SNR of the simulation signal compared the the estimated SNRs when taken from the frequency domains of FFT and envelope analysis, and the quefrency domain from cepstrum analysis. The figure shows the SNRs from the frequency/quefrency domains follow a similar shape as the magnitude of the Gaussian noise is increased.

This simple estimate of the SNR from the frequency domain provides a means to assess the signal before and after the wavelet de-noising process. Comparing the before and after SNR provides information about the effect the de-noising step has on the vibration signal.

5.3 Simulation Signal - Single Data Frame

The literature review identified three common methods for extracting CFFs. The methods are based on standard Fourier analysis, cepstrum analysis, and envelope analysis. The first objective when developing these methods was to analyse a single frame of data and produce accurate results. At this early stage of development each extraction method was created independent of each other with the idea that in the future all three would combine into a single program that could analyse a data file simultaneously.

The first step was to get very basic versions of these methods working to better understand

the respective program's requirements in terms of input file size, sample frequency, frame length, frequency/quefrency axis, and also to observe the output from basic input signals. The Fourier analysis method was created first. A 50 Hz continuous sine wave was used as the input signal which produced the standard 50 Hz frequency component. I then started to space single 50 Hz sine wave cycles apart from one another by adding zero signal between cycles. The purpose of this was to observe the response in the frequency spectrum when the input signal was made up of periodic pulses. In the frequency spectrum the 50 Hz component was still the main component however additional beat frequencies began to enter the spectrum surrounding the 50 Hz component. Next the basic cepstum analysis was developed. The literature review had explained that cepstrum analysis excelled at revealing periodicities in a signal so the first basic input signal was a very short square wave that repeated every 20 ms. The result was a very strong quefrency component exactly at 20 ms. The third extraction method to begin developing was envelope analysis. During the literature review I discovered work by Bechhoefer (2012) that provided an introduction to envelope analysis specifically for bearing vibration signals. The work detailed the function of envelope analysis and also provided MATLAB^{TM} code for conducting bearing vibration analysis. Using the programs provided by Bechhoefer (2012) I tested the envelope analysis method using some of the real vibration data files I had acquired. The results were very accurate so the decision was made that no further development of envelope analysis was required until future stages where I would combine the three extract methods into a single program.

Returning to Fourier and cepstrum analysis the next step was to test the programs using the simulation signal I had developed (refer to Chapter 4 for details of the simulation signal used). The main frequency component within the simulation signal was 1666 Hz and the periodicity of the cosine pulse was 20 ms. When processing the simulation signal with the Fourier and cepstrum methods the main frequency component was 1611 Hz and the main quefrency component was 20.05 ms. The difference in the cepstrum quefrency component was accepted. The difference in the frequency component from 1666 Hz to 1611 Hz was due to the frame length being analysed in the Fast Fourier Transform (FFT) process. The first iteration of the simulation signal was only 4000 samples long which meant the maximum frame size that could be analysed was only 2048 samples.

The development of the Fourier and cepstrum methods moved on to analysing the simulation signal with varying levels of Gaussian noise added. The 4000 sample simulation

signal was converted into 10 signals each with a different level of Gaussian noise added (later revised to nine magnitudes of Gaussian noise), and the SNR for each simulation signal was calculated using the traditional method of SNR. The signals were processed using the Fourier and cepstrum analysis methods. For signals with stronger SNR the peak frequency/quefrency components matched the expected results of 1611 Hz and 20.05 ms. However, for the signals where the magnitude of Gaussian noise was larger and the SNR was lower the peak frequency/quefrency components appeared to be random and did not align with the noise free simulation signal components. This result was to be expected. I assumed as the SNR was reduced the difficulty in extracting the main frequency/quefrency components would increase. I also expected that wavelet de-noising would remove the noise from the simulation signal in such a way that the 1611 Hz and 20.05 ms components would again become dominant.

I de-noised the low SNR simulation signals using a variety of wavelet de-noising parameters. In almost all cases the de-noising process did not change the peak frequency/quefrency component. To try and understand why the de-noising was not producing the results I expected I attempted to quantify the effect of the wavelet de-noising process by measuring the simulation signal SNR before and after the de-noising step. During the testing of this quantifying process I wrote a program that created a new noise vector to add to the noise free simulation signal each time the program was run. As each new noise vector was added to the simulation signal I observed the peak frequency and quefrency components changing.

The conclusion I drew from these results was that the frequency content within the short 4000 sample Gaussian noise vector was random each time the vector was created. When this frequency content was combined with the simulation signal frequency content the peak frequency/quefrency components were also random from one frame to the next. Processing single frames of data in this way was not to going to be suitable to accurately describe the effect of the wavelet de-noising process.

5.4 Simulation Signal - Multiple Data Frames

The process of analyisng a single frame of data individually produced random results and made assessing the effect of the wavelet de-noising process difficult. In an attempt to

smooth out and average the results I decided to process the simulation signal 1000 times, each with an independent noise vector added. I calculated the SNR using the traditional method for each frame and then took the average which provided a more meaningful and consistent measure of the true SNR with respect to the magnitude of Gaussian noise being added. I then included a wavelet de-noising process for each of the 1000 data frames and calculated the average SNR after the de-noising step. At this stage in the development I was using the Traditional SNR calculation described above under Section 5.2.1. The averaging of the SNR using 1000 frames of data provided consistent, repeatable results so I then progressed onto testing all combinations of wavelet de-noising parameters.

The literature review provided some guidance on what wavelet types to use when de-noising vibration signals, recommending Daubechies, Myer and Morlet wavelet families. In this work I have used the Daubechies family from DB1 to DB19 and I initially set the maximum level of decomposition to 20 levels. As has been discussed in the literature review chapter there are many de-noising parameters to chose from however the literature does not provide a great deal of advice when selecting parameters specifically for de-noising vibration signals. To set some limits when testing the broad range of de-noising parameters I decided to utilise the range of inputs available for the MATLAB^{TM} function *wden()*.

A summary of the range of wavelet de-noising parameter combinations is provided:

- 4 threshold types.
- 2 means of applying the threshold.
- 3 types of noise estimation methods.
- 19 different Daubechies wavelets.
- 20 levels of wavelet decomposition.

This range of de-noising parameters produces 9120 unique wavelet de-noising combinations.

			Decomposition level 1			
			RigrSURE	SqTwoLog	HeurSURE	Minimaxi
DB1	Soft	One				
		Sln				
		Mln				
	Hard	One				
		Sln				
		Mln				

Figure 5.6: An example of the matrix arrangement for storing the SNR changes.

I modified my Fourier and cepstrum analysis programs to include the above wavelet de-noising combinations. Envelope analysis was still left out of development at this stage of the project. The Fourier and cepstrum programs followed the same algorithm. Each program loaded the simulation signal, processed 1000 frames of noisy data where the added Gaussian noise vector used the same magnitude of noise, and the average SNR over the 1000 frames was taken. The 1000 noisy frames were then wavelet de-noised using one combination of de-noising parameters, and the new average SNR was calculated again. The new SNR was subtracted from the original SNR and the difference stored in a matrix that correlated to the combination of wavelet de-noisig parameters used. Figure 5.6 shows the matrix arrangement for storing the SNR changes resulting from the de-noising parameter combinations for the 1st level of decomposition and DB1. One wavelet de-noising parameter was changed and then the original 1000 noisy frames were de-noised again, and so on. The product of this process was a single matrix that stored the change in SNR produced by the 9120 possible combinations of wavelet de-noising parameters (4 thresholds, 3 noise scale estimations, 2 applications of threshold, 20 levels of wavelet decomposition and 19 wavelets from the Daubechies family). The program then changed the magnitude of the Gaussian added noise and ran again until 10 levels of Gaussian added noise had been analysed.

The results of the programs were analysed. Within the Fourier analysis results it was identified that some unusually large gains in SNR were occurring from wavelet de-noising combinations that used decomposition levels of 4 and above. The reason for these large gains in SNR have been discussed above under Section 5.2.1. The de-noised signals were checked manually using Fourier anslysis and it was clear the only levels of wavelet decomposition that were not severely distorting the signal were levels 1, 2 and 3, although for all level 3 cases the de-noised SNR was much poorer than the original. On this basis further testing and assessment of simulation signals would only use wavelet decomposition levels up to and including level 3.

After considering the next step in the development of the programs I realised the existing method for assessing the improvement in signal quality from the wavelet de-noising process could not be applied to the real vibration signal datasets as there was no means to measure the signal or noise strength independently (refer Section 5.2.1). Using the knowledge that the Gaussian noise vector changed the resulting peak frequency/quefrency component in a random way I decided to test the idea that the peak frequency/quefrency components

might be distributed around the desired 1611 Hz and 20.05 ms components. I adjusted the Fourier and cepstrum programs to record the peak components for each of the 1000 frames processed and plotted a histogram of the results. It was clear that although the peak components were not always the desired 1611 Hz and 20.05ms the incorrect peaks were distributed around the correct components. Section 5.2.2 above discusses this concept further. Before re-processing the simulation signals using the new method for assessing the effect of wavelet de-noising I began testing the technique on real vibration signals.

5.5 Simulation Signal - SNR in the Frequency Domain

To complete the discussion of analysis of the simulation signals I will explain the final process decided upon for analysing the simulation signals. Note that presenting this section here is out of logical sequence with the true order of events and skips a step in the development of the method quantifying signal improvement after de-noising. The correct sequence for quantifying the signal improvement after de-noising was explained in Section 5.2.

Ultimately the simulation signals were the last set of signals to be processed for this project. The simulation signals had served their initial purpose of developing and calibrating the CFF extraction methods and when I started to process real vibration signals I left the simulation signals to last. By this stage the method for calculating SNR within the frequency domain had been developed and so too had the final envelope analysis method. Envelope analysis of the simulation signal appeared to extract a 50 Hz component from the frequency content that made up the damped oscillation pulse within the simulation signal. The 50 Hz component relates to the fact that the oscillation pulse occurred every 20 ms, or at 50 Hz. The low frequency 50 Hz signal is being demodulated from the higher frequency content, in the range of 1611 Hz (including the surrounding beat frequencies). When analysing the simulation signal I set the envelope analysis target frequency to be 50 Hz.

All three CFF extraction methods were fully functional and the full set of simulation signals were processed.

5.6 Real Vibration Signals

5.6.1 Short Time Signals

In the preceding section, and also in Section 5.2.2 I discussed the idea of assessing the effect of the wavelet de-noising process by counting how many times out of 1000 the peak frequency/quefrency component was correctly selected. Before I re-processed the simulation signals using this method I decided I should check to see if this method was suitable for use on the real vibration signals. This method was found to be unsuitable for use on the testbed vibration signals and the reasons for this are discussed in Section 5.2.2.

Some further research revealed a possible method for estimating the SNR from within the frequency domain. This process has been discussed in Section 5.2.3 and was extended to apply to the quefrency domain when conducting cepstrum analysis. With a new method for quantifying the effect of the wavelet de-noising process I then began process the Short Time Signals which were all acquired from a bearing testbed arrangements. The first step towards this objective was to combine all three CFF extract methods into a single program that could run autonomously and process vibration files in batches. At this point in the project I revisited the envelope detection method with the intention of including it in the automated Fourier and cepstrum analysis programs.

The original envelop detection method provided by Bechhoefer (2012) required the user to analyse a graph and decide which frequency band contained the bearing high frequency resonance. This requirement for user interaction at the initial stage of the program analysis prevented complete autonomous running. I decided this would not suit my requirements and searched for an alternative envelope detection method. A brief search of the literature indicated a similar envelope analysis method could be obtained using a Hilbert transform. The Hilbert transform also features as a MATLAB^{TM} function and could be combined with the already written Fourier analysis program to form the envelope analysis component. Essentially the input signal is Hilbert transformed and then analysed in the same manner as the Fourier analysis.

With a new method for envelope analysis established I combined all three CFF extraction methods into a single program. I tested the new program using real vibration signals to re-assess the maximum level of wavelet decomposition that should be used for the real

bearing vibration files. As discussed above when analysing the simulation signals, wavelet decomposition levels above 4 produced problematic results. The re-test conducted at this stage indicated the wavelet decomposition level could be set to 10 levels without experiencing problems. The program followed a similar algorithm as earlier and a flowchart of the algorithm is shown in Figure 5.7. Due to the short length of the real vibration signals, often only one or a few seconds in duration, I could no longer analyse 1000 frames separately. I reconfigured the program to analyse 10 frames of data in total and spread these frames evenly throughout the duration of the signal.

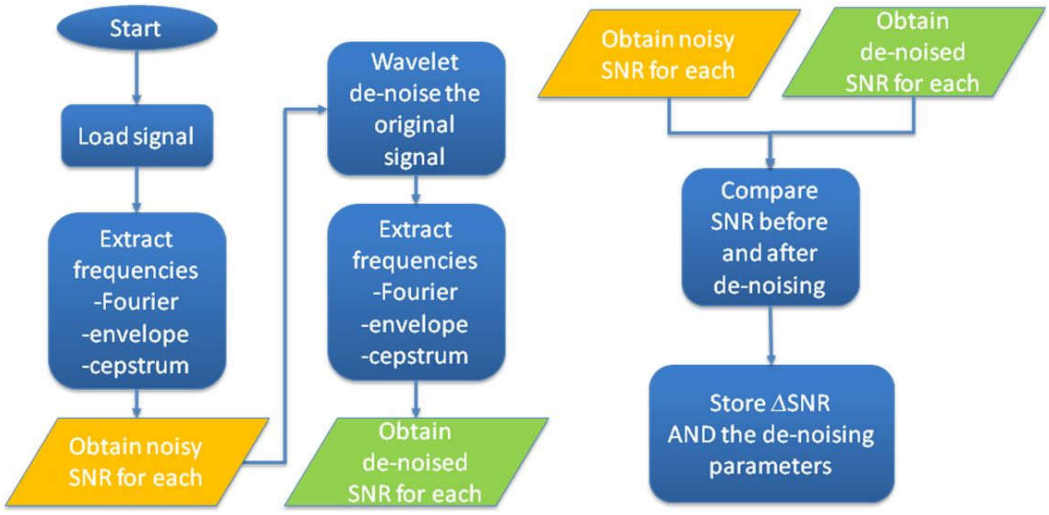

Figure 5.7: The flowchart shows the algorithm for processing the vibration data files.

A total of 70 short time vibration data files were analysed and each file analysed produced an output file. Contained within each output file is three matrices, one for Fourier analysis, cepstrum analysis and envelope analysis. Since the level of wavelet decomposition was set to level 10 for the short time signals, each matrix has the SNR results from 4560 different combinations of wavelet de-noising parameters. 70 files each with 3 matrices that each hold 4560 SNR values equates to a total of 957,600 wavelet de-noising parameter combinations assessed using the short time signals.

5.6.2 Long Time Signals

Long time signals consist of a sequence of bearing vibration data files that record the vibration profile of a bearing over a period of days or weeks. The files analysed in this stage of the project were sourced from a bearing testbed arrangement that monitored and recorded bearing vibrations every 10 minutes over a period of 32 days and provided an insight into the vibration signature of the bearing at varying levels of wear, from brand new to failure. Each data file recording was 1 second long. I decided to analyse one data file captured on each day of the experiment totaling 32 files. I extracted and grouped the 32 files and renamed them to allow easier processing. The main analysis program was structured in a way which allowed batch processing so all 32 files were processed at once albeit sequentially. Each file produced an output file in the same manner as the short time signals.

5.7 Post Processing of Results

The product of the analysis of all three signal groups (simulation, short time and long time signals) was a large collection of matrices which had stored the change in SNR relating to a particular wavelet de-noising parameter combination. The next step was to condense this wide range of results into information that could be more easily understood and interpreted. The same post processing method was used for all three signal groups however for the short time signals there was an intermediate step where I manually grouped the results into four sub-groups relating to the type of bearing fault being analysed. The four sub-groups were rolling element faults, inner race faults, outer race faults and normal operating bearings.

The first program I wrote loaded a *results* file produced from the above program and extracted a matrix relating to one of the CFF extraction methods. Each matrix had 4560 SNR changes produced by the range of wavelet de-noising parameter combinations. The program extracted all the SNR changes relating to a wavelet de-noising variable, summed and averaged the data and then graphed the results against the level of wavelet decomposition. This produced three graphs, one for the type of threshold, one for the application of threshold and one for the noise scale estimation method. An example of each graph produced is provided in Figures 6.1, 6.2 and 6.3. The condensed results for

the wavelet de-noising variables were also stored in another matrix and saved.

The program also calculated the average SNR gain resulting from each Daubechies wavelet and graphed the results. An example of the graph is shown in Figure 6.4.

For the simulation signals and the long time signals the graphs produced at this stage were sufficient for conducting further analysis. The short time signals however still contained a large amount of data that required further condensing. A secondary program was written that summarised the results of the four sub-groups described above. Graphs were produced summarising the wavelet de-noising parameters for rolling element faults, inner race faults, outer race faults and normal operating bearings. These graphs appear in Chapter 6. These results were further summarised to compare the overall best performing thresholds, noise scale estimations, threshold applications and Daubechies wavelet from the short time signals analysis.

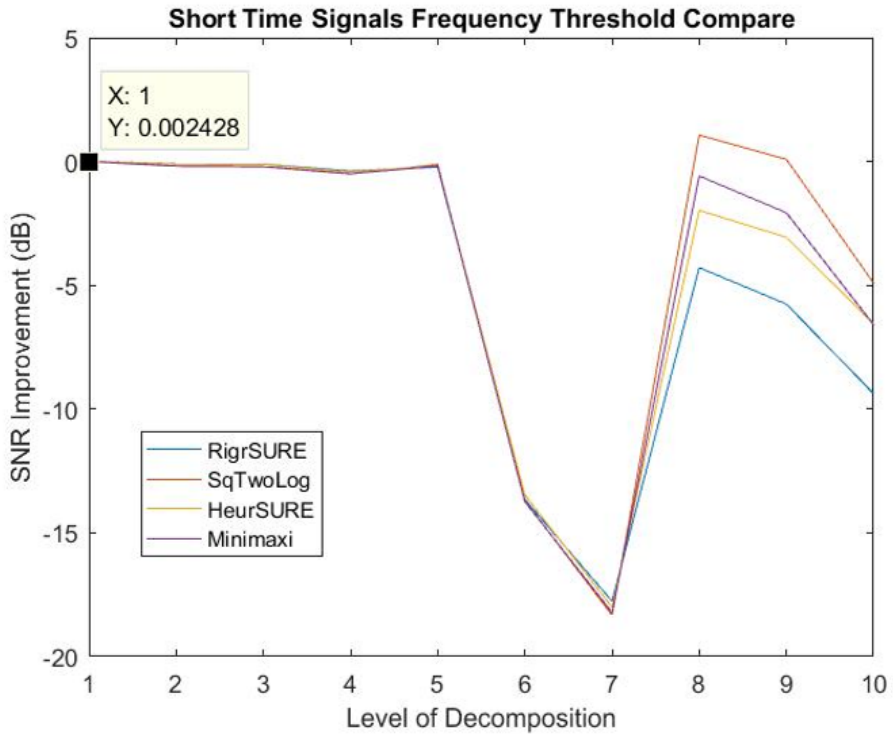

Figure 5.8: An example of the threshold comparison graph.

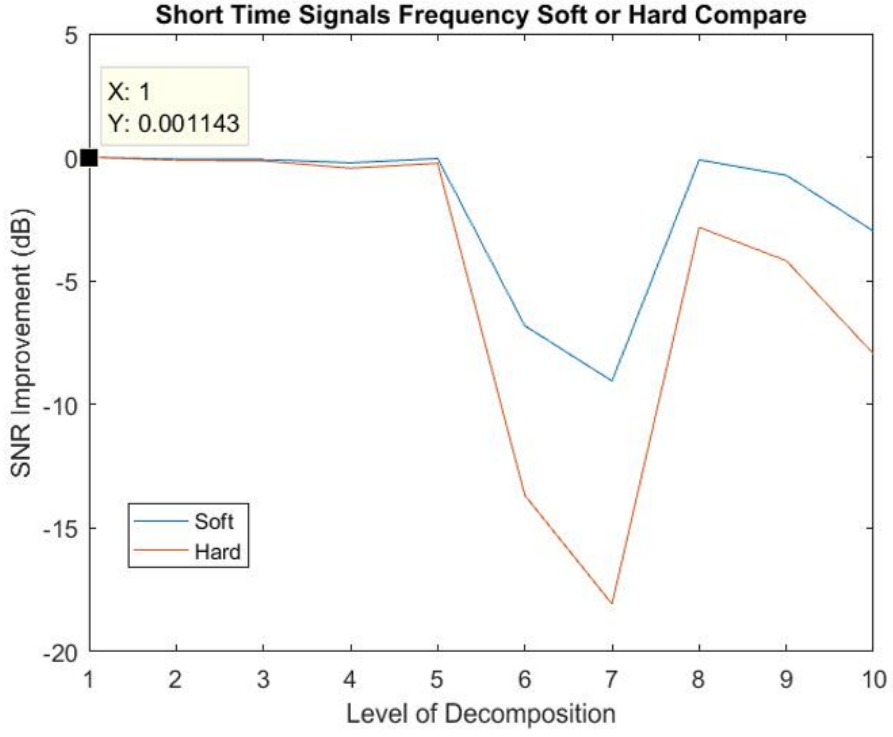

Figure 5.9: An example of the soft or hard comparison graph.

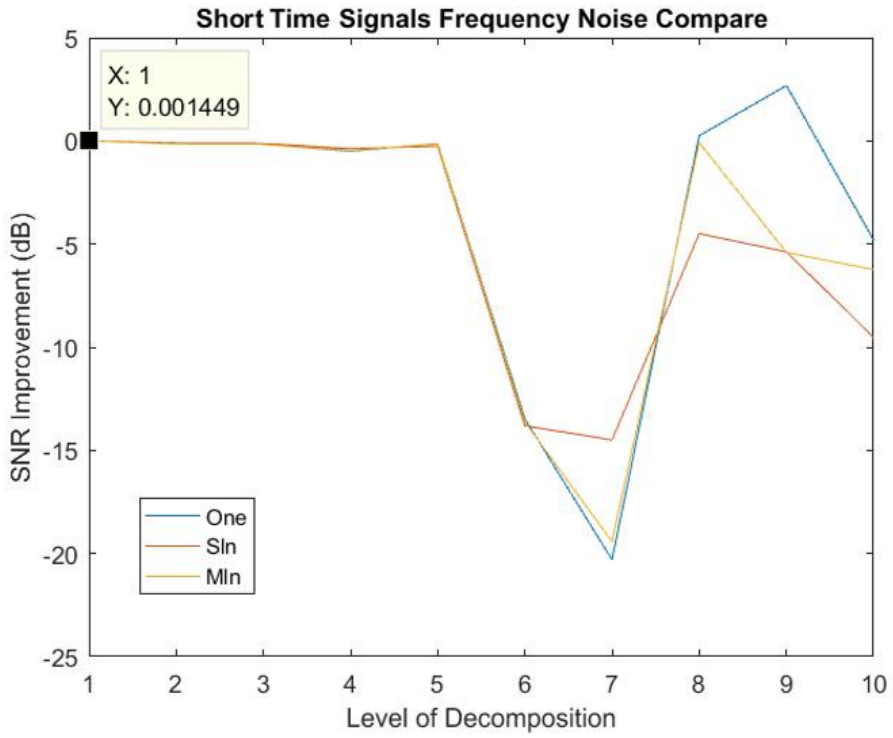

Figure 5.10: An example of the noise scale comparison graph.

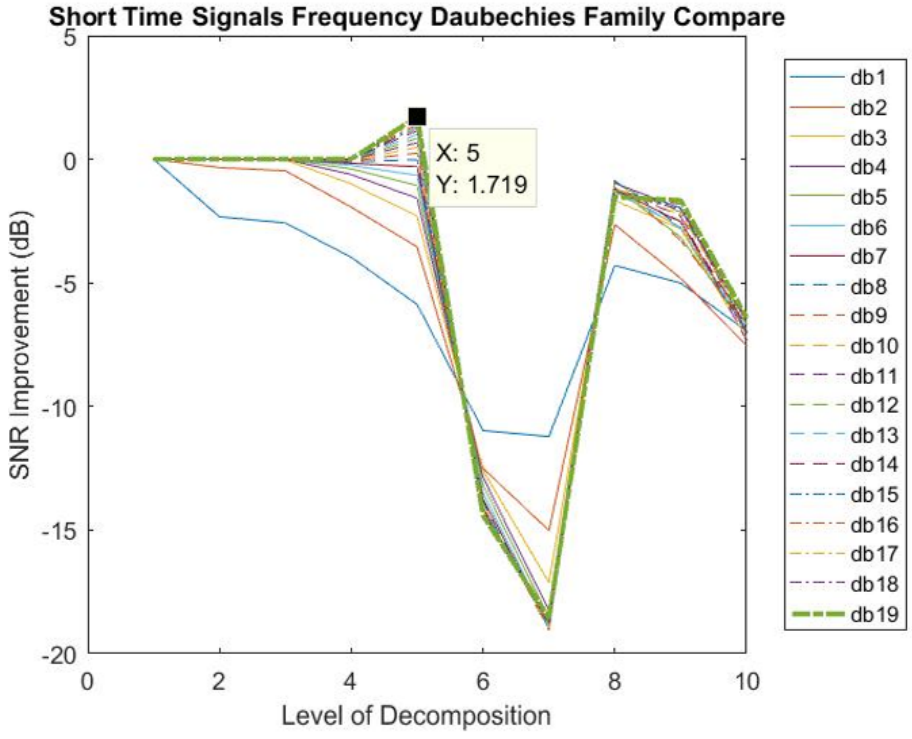

Figure 5.11: An example of the Daubechies family comparison graph.

Chapter 6

Results

6.1 Chapter Overview

6.2 Simulation Signal

The simulation signal results presented here have been summarised and tabulated rather than graphed. It was deemed impractical to present all 12 graphs for each of the nine simulation files analysed.

Tables 6.1, 6.2, 6.3 and 6.4 present the best performing wavelet de-noising parameters when the simulation signal was analysed using Fourier analysis.

Table 6.1: The best performing wavelet de-noising thresholds using Fourier analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Threshold**	**SNR improvement**
0.0	1	All thresholds	0.00 dB
0.1	1	RigrSURE	-0.03 dB
0.2	1	All thresholds	-0.06 dB
0.3	1	All thresholds	-0.08 dB
0.4	1	All thresholds	-0.10 dB
0.5	1	RigrSURE	-0.09 dB
0.6	1	RigrSURE	-0.09 dB
0.7	1	Minimaxi	-0.08 dB
0.8	1	Minimaxi	-0.08 dB

Table 6.2: The best performing application of wavelet de-noising thresholds (soft or hard application) using Fourier analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Soft or Hard**	**SNR improvement**
0.0	1	Soft	0.00 dB
0.1	1	Soft	-0.01 dB
0.2	1	Soft	-0.03 dB
0.3	1	Soft	-0.04 dB
0.4	1	Soft	-0.05 dB
0.5	1	Soft	-0.05 dB
0.6	1	Soft	-0.04 dB
0.7	1	Soft	-0.04 dB
0.8	1	Soft	-0.03 dB

Table 6.3: The best performing wavelet de-noising noise estimation method using Fourier analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Noise estimation**	**SNR improvement**
0.0	1	Mln and Sln	0.00 dB
0.1	1	Mln and Sln	-0.03 dB
0.2	1	Mln and Sln	-0.06 dB
0.3	1	Mln and Sln	-0.08 dB
0.4	1	All methods	-0.10 dB
0.5	1	Mln and Sln	-0.09 dB
0.6	1	Mln and Sln	-0.09 dB
0.7	1	One	-0.08 dB
0.8	1	Mln and Sln	-0.08 dB

Table 6.4: The best performing Daubechies wavelet for de-noising using Fourier analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Daubechies wavelet**	**SNR improvement**
0.0	2	DB11	0.04 dB
0.1	2	DB11	0.08 dB
0.2	2	DB10	0.05 dB
0.3	2	DB11	0.06 dB
0.4	2	DB14	0.00 dB
0.5	2	DB11	0.05 dB
0.6	2	DB9	0.01 dB
0.7	2	DB10	0.00 dB
0.8	2	DB8	0.05 dB

Tables 6.5, 6.6, 6.7 and 6.8 present the best performing wavelet de-noising parameters when the simulation signal was analysed using envelope analysis.

Table 6.5: The best performing wavelet de-noising thresholds using envelope analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Threshold	SNR improvement
0.0	7	SqTwoLog	19.09 dB
0.1	7	SqTwoLog	26.03 dB
0.2	7	SqTwoLog	29.67 dB
0.3	7	SqTwoLog	31.12 dB
0.4	7	SqTwoLog	32.33 dB
0.5	7	SqTwoLog	32.50 dB
0.6	7	SqTwoLog	32.79 dB
0.7	7	SqTwoLog	31.96 dB
0.8	7	SqTwoLog	31.87 dB

Table 6.6: The best performing application of wavelet de-noising thresholds (soft or hard application) using envelope analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Soft or Hard	SNR improvement
0.0	7	Hard	17.60 dB
0.1	7	Hard	18.29 dB
0.2	7	Hard	20.53 dB
0.3	7	Hard	22.47 dB
0.4	7	Hard	24.19 dB
0.5	7	Hard	24.75 dB
0.6	7	Hard	25.20 dB
0.7	7	Hard	24.92 dB
0.8	7	Hard	24.42 dB

Table 6.7: The best performing wavelet de-noising noise estimation method using envelope analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Noise estimation	SNR improvement
0.0	7	One	44.35 dB
0.1	7	One	42.82 dB
0.2	7	One	37.40 dB
0.3	7	One	34.44 dB
0.4	7	One	34.18 dB
0.5	7	One	33.81 dB
0.6	7	One	33.83 dB
0.7	7	One	32.83 dB
0.8	7	One	32.82 dB

Table 6.8: The best performing Daubechies wavelet for de-noising using envelope analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Daubechies wavelet	SNR improvement
0.0	7	DB18	26.66 dB
0.1	7	DB18	29.53 dB
0.2	7	DB18	28.52 dB
0.3	7	DB18	27.53 dB
0.4	7	DB14	27.80 dB
0.5	7	DB15	28.18 dB
0.6	7	DB12	28.21 dB
0.7	7	DB17	27.40 dB
0.8	7	DB13	27.22 dB

Tables 6.9, 6.10, 6.11 and 6.12 present the best performing wavelet de-noising parameters when the simulation signal was analysed using cepstrum analysis.

Table 6.9: The best performing wavelet de-noising thresholds using cepstrum analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Threshold**	**SNR improvement**
0.0	1	All thresholds	-0.05 dB
0.1	3	SqTwoLog	-6.07 dB
0.2	3	HeurSURE	1.03 dB
0.3	2	SqTwoLog	5.81 dB
0.4	2	SqTwoLog	7.43 dB
0.5	2	HeurSURE	7.09 dB
0.6	2	SqTwoLog	5.57 dB
0.7	2	Minimaxi	5.01 dB
0.8	2	SqTwoLog	3.23 dB

Table 6.10: The best performing application of wavelet de-noising thresholds (soft or hard application) using cepstrum analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	**Level of wavelet decomposition**	**Soft or Hard**	**SNR improvement**
0.0	1	Soft	-0.03 dB
0.1	3	Soft	-3.50 dB
0.2	3	Soft	-0.06 dB
0.3	2	Hard	3.72 dB
0.4	2	Hard	5.49 dB
0.5	2	Hard	4.67 dB
0.6	2	Hard	4.93 dB
0.7	2	Hard	4.48 dB
0.8	2	Hard	1.96 dB

Table 6.11: The best performing wavelet de-noising noise estimation method using cepstrum analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Noise estimation	SNR improvement
0.0	5	Mln	1.10 dB
0.1	4	Mln	-4.79 dB
0.2	4	Sln	2.13 dB
0.3	2	One	5.81 dB
0.4	2	One	7.43 dB
0.5	2	One	7.12 dB
0.6	2	One	5.57 dB
0.7	2	Sln	4.89 dB
0.8	2	One	3.19 dB

Table 6.12: The best performing Daubechies wavelet for de-noising using cepstrum analysis, applied to the simulation signal.

Magnitude of added Gaussian noise	Level of wavelet decomposition	Daubechies wavelet	SNR improvement
0.0	5	DB2	0.27 dB
0.1	7	DB1	-2.25 dB
0.2	3	DB2	2.70 dB
0.3	3	DB11	6.66 dB
0.4	3	DB4	9.35 dB
0.5	8	DB4	10.66 dB
0.6	2	DB10	7.19 dB
0.7	2	DB11	10.55 dB
0.8	3	DB5	6.65 dB

6.3 Short Time Signals - Testbed Vibration Data

6.3.1 Short Time Signals - All Combined

The following graphs provide the average SNR improvement as a result of wavelet de-noising for all short time signals analysed. Figures 6.1, 6.2, 6.3 and 6.4 summarise the signal to noise ratio (SNR) improvement when Fourier analysis was used to extract the characteristic fault frequencies (CFFs). Figures 6.5, 6.6, 6.7 and 6.8 summarise the SNR improvement when envelope analysis was used to extract the CFFs. Figures 6.9, 6.10, 6.11 and 6.12 summarise the SNR improvement when cepstrum analysis was used to extract the CFFs. On the graphs displaying the Daubechies family comparison the most effective Daubechies wavelet is boldly coloured.

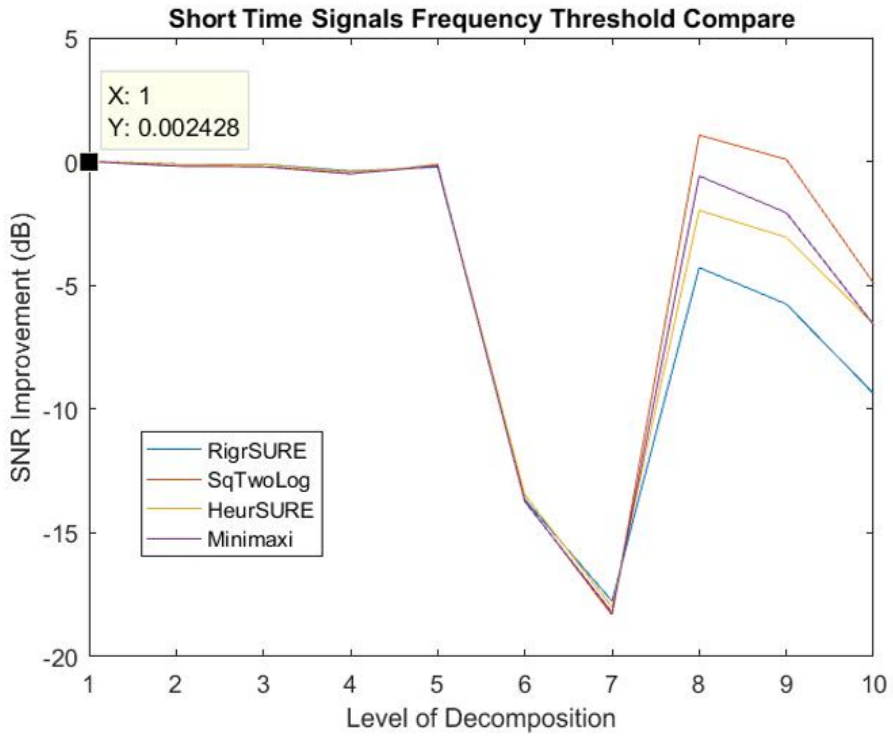

Figure 6.1: Short Time Signals - Fourier analysis threshold comparison.

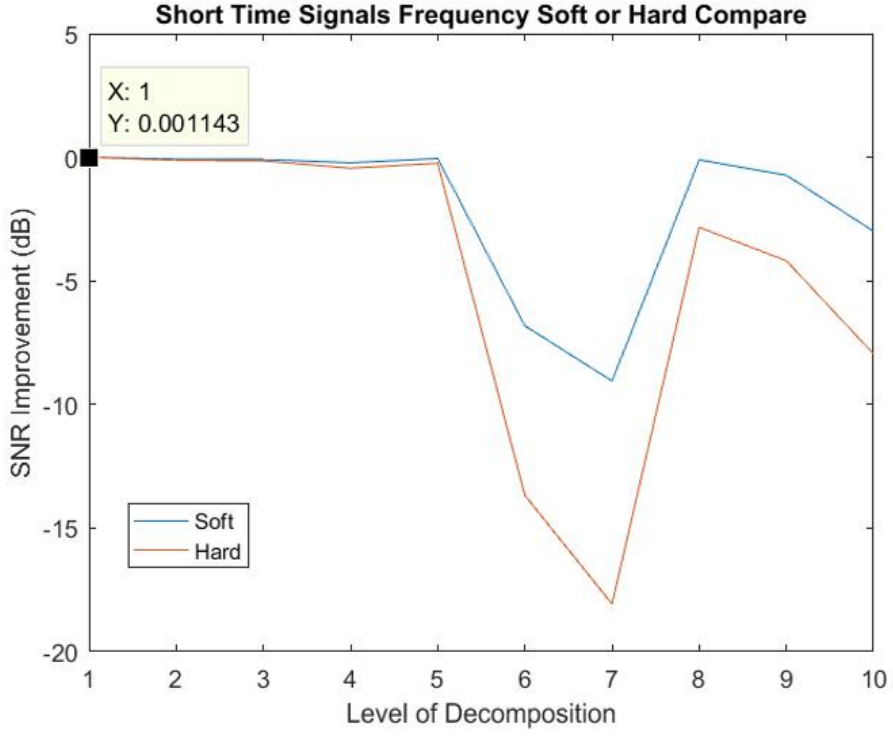

Figure 6.2: Short Time Signals - Fourier analysis threshold application comparison.

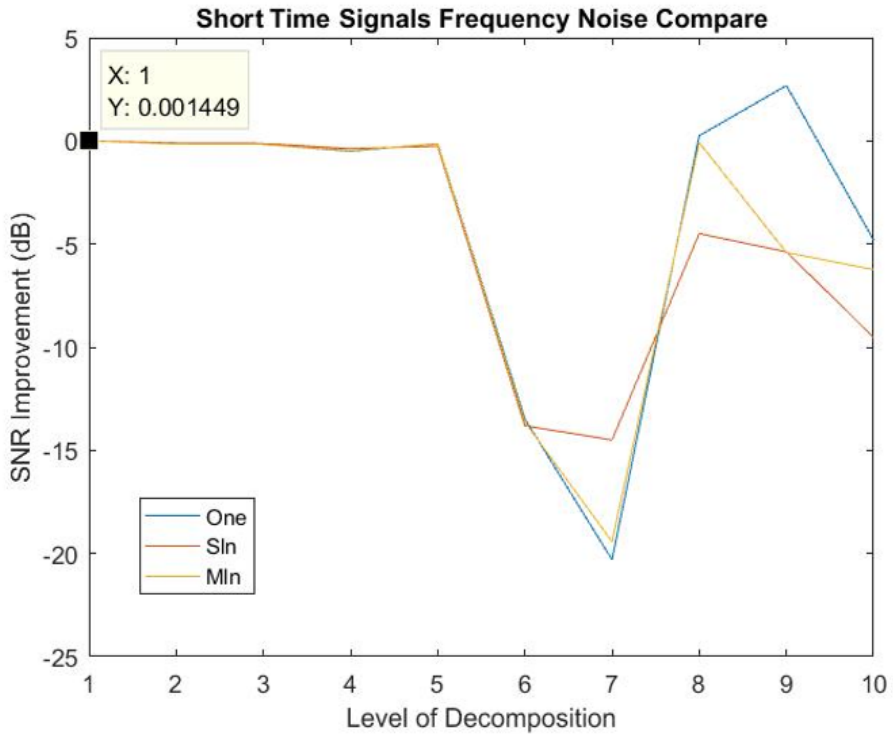

Figure 6.3: Short Time Signals - Fourier analysis noise scale estimation comparison.

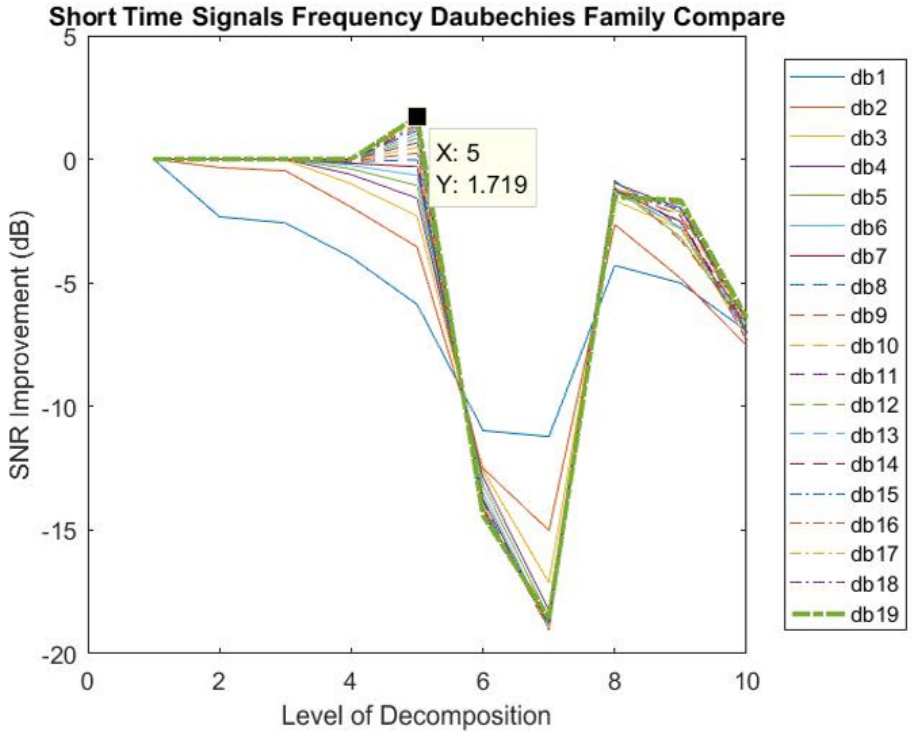

Figure 6.4: Short Time Signals - Fourier analysis Daubechies family comparison.

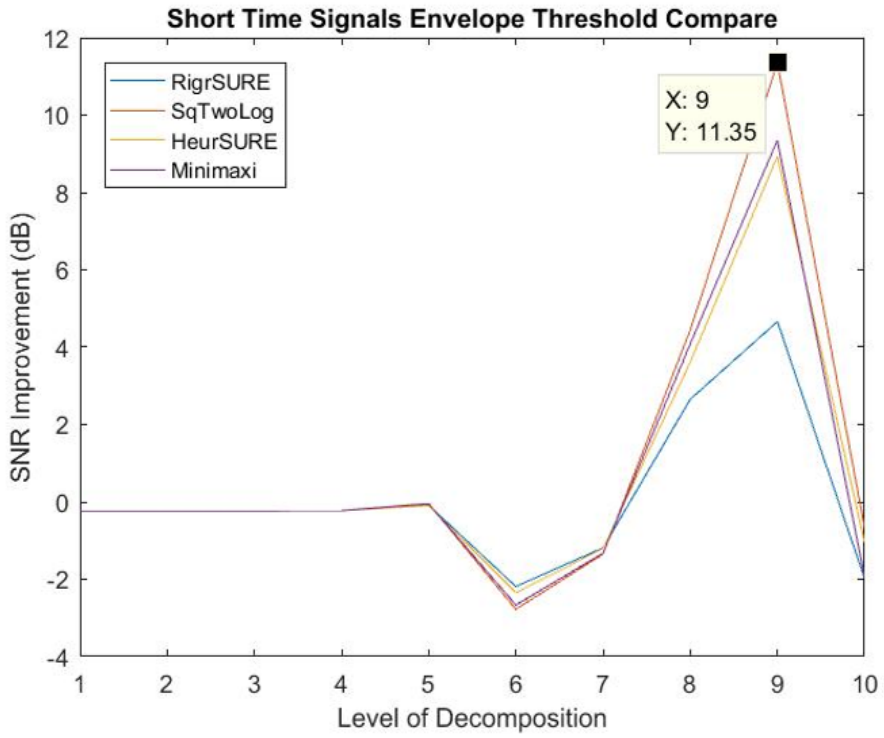

Figure 6.5: Short Time Signals - envelope analysis threshold comparison.

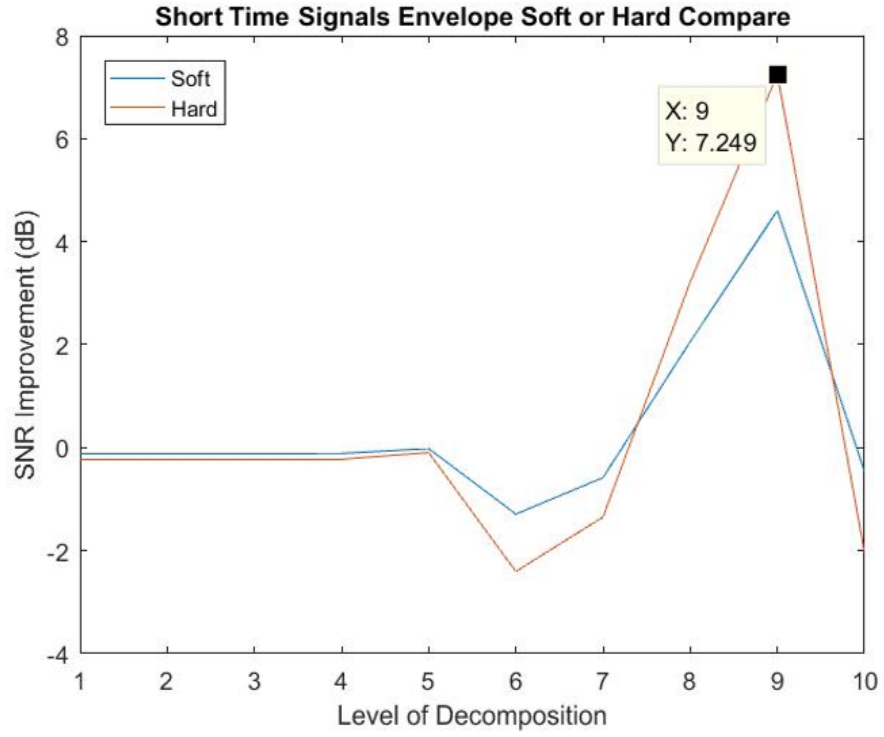

Figure 6.6: Short Time Signals - envelop analysis threshold application comparison.

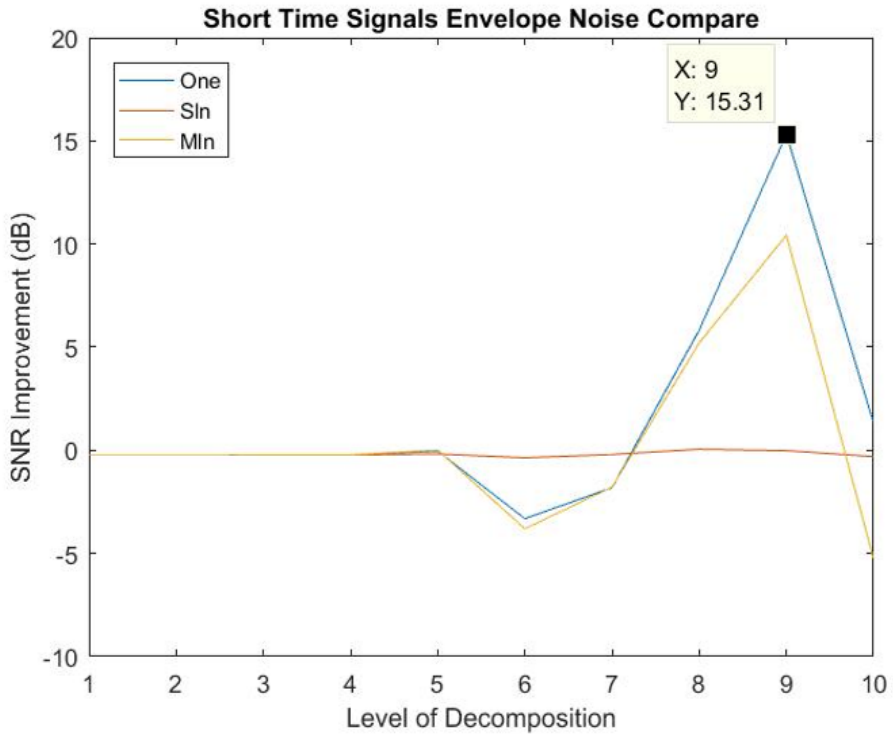

Figure 6.7: Short Time Signals - envelope analysis noise scale estimation comparison.

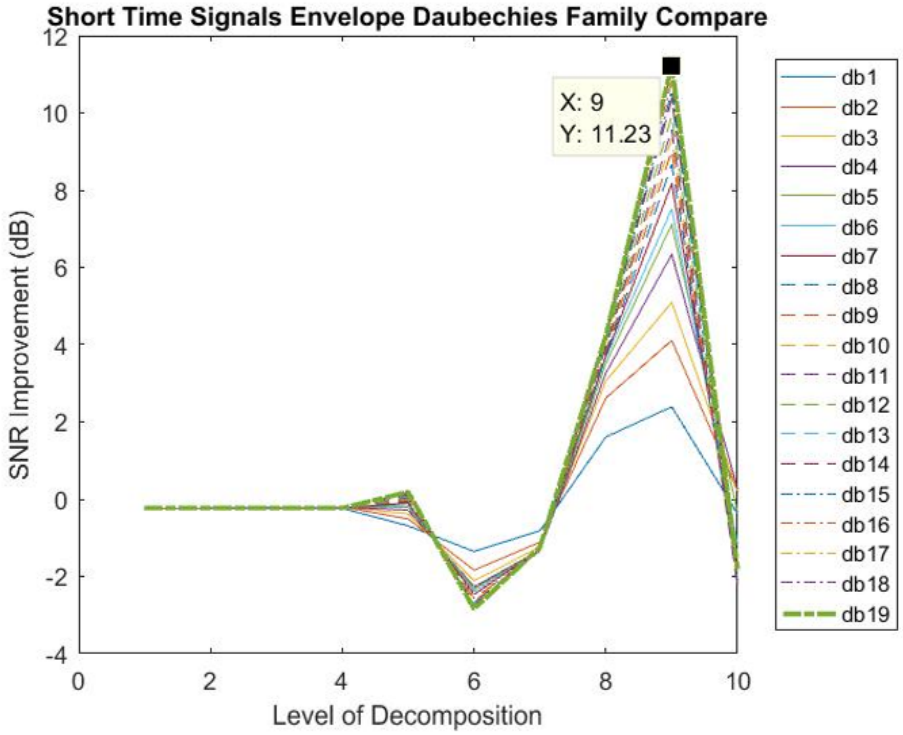

Figure 6.8: Short Time Signals - envelope analysis Daubechies family comparison.

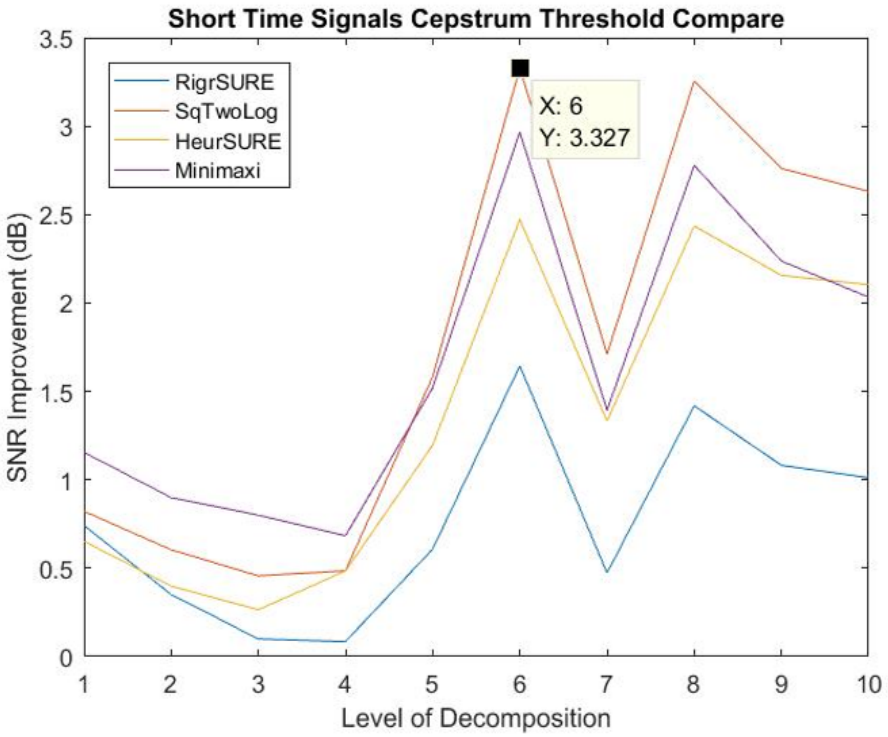

Figure 6.9: Short Time Signals - cepstrum analysis threshold comparison.

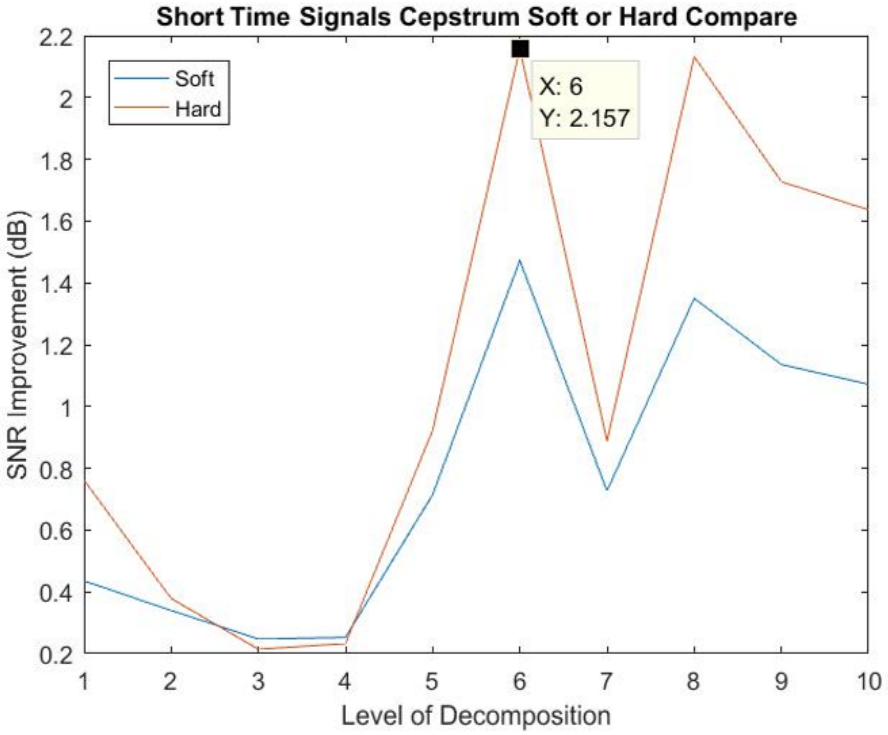

Figure 6.10: Short Time Signals - cepstrum analysis threshold application comparison.

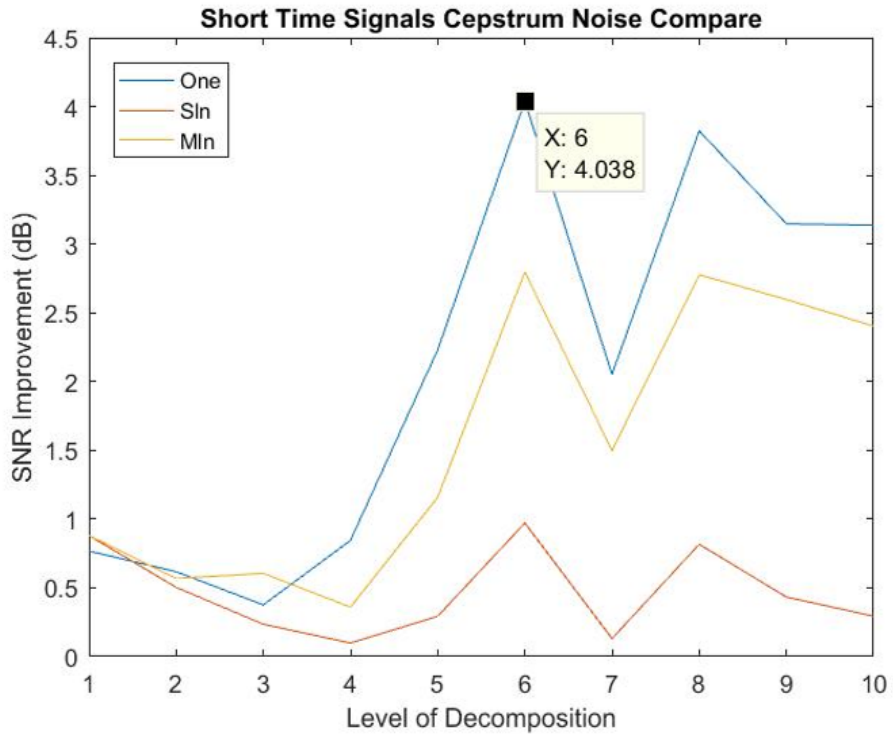

Figure 6.11: Short Time Signals - cepstrum analysis noise scale estimation comparison.

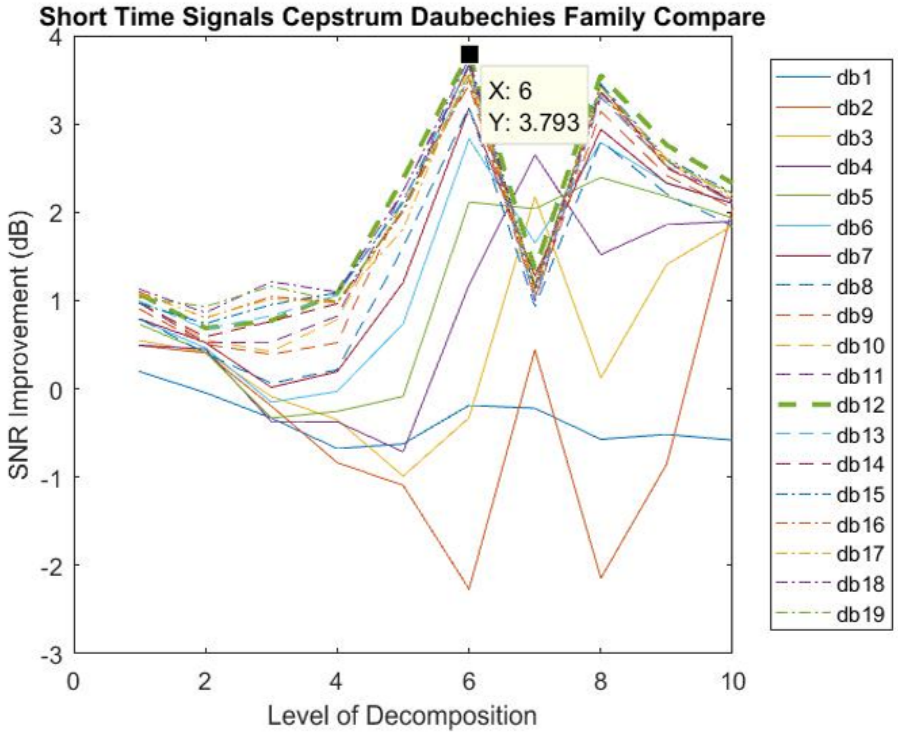

Figure 6.12: Short Time Signals - cepstrum analysis Daubechies family comparison.

6.3.2 Short Time Signals - Normal Bearing

The bearing vibration datasets from Case Western Reserve Bearing Data Centre and Machinery Failure Prevention Technology group provide vibration signals recorded from bearings with no failure mode present.

The following graphs provide the average SNR improvement as a result of wavelet de-noising for bearings with no failure mode present. Seven files were analysed and the results averaged. Figures 6.13, 6.14, 6.15 and 6.16 summarise the SNR improvement when Fourier analysis was used to extract the CFFs. Figures 6.17, 6.18, 6.19 and 6.20 summarise the SNR improvement when envelope analysis was used to extract the CFFs. Figures 6.21, 6.22, 6.23 and 6.24 summarise the SNR improvement when cepstrum analysis was used to extract the CFFs. On the graphs displaying the Daubechies family comparison the most effective Daubechies wavelet is boldly coloured.

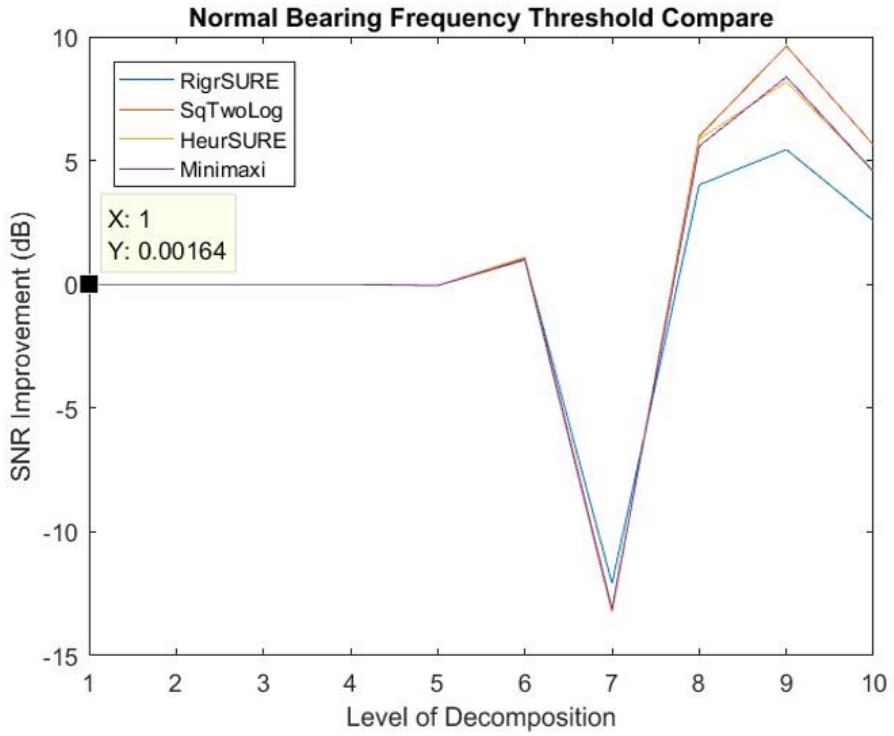

Figure 6.13: Short Time Signals - Fourier analysis, normal bearing threshold comparison.

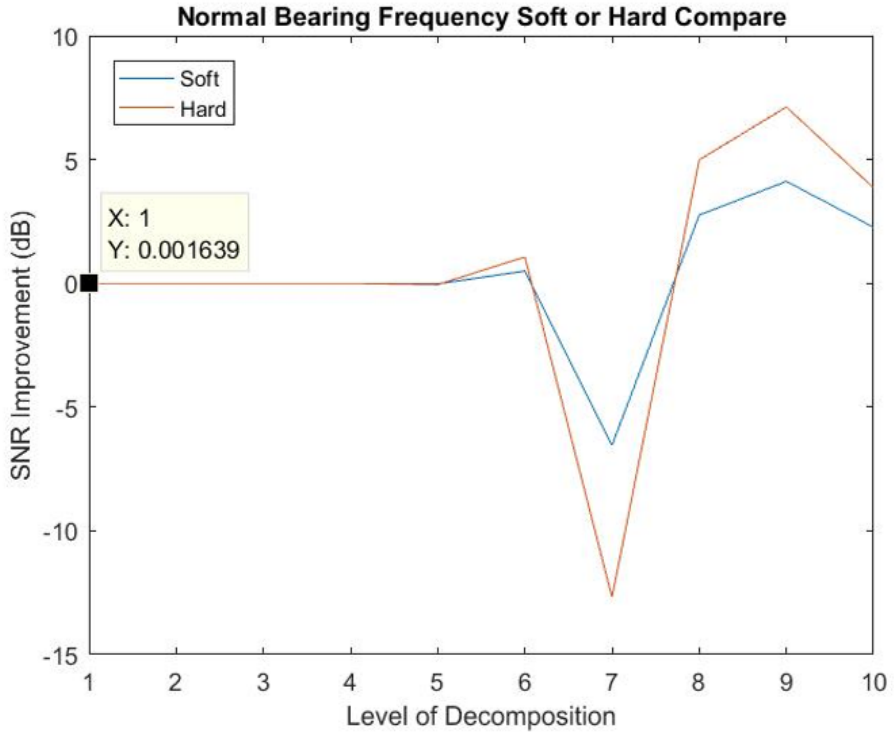

Figure 6.14: Short Time Signals - Fourier analysis, normal bearing threshold application comparison.

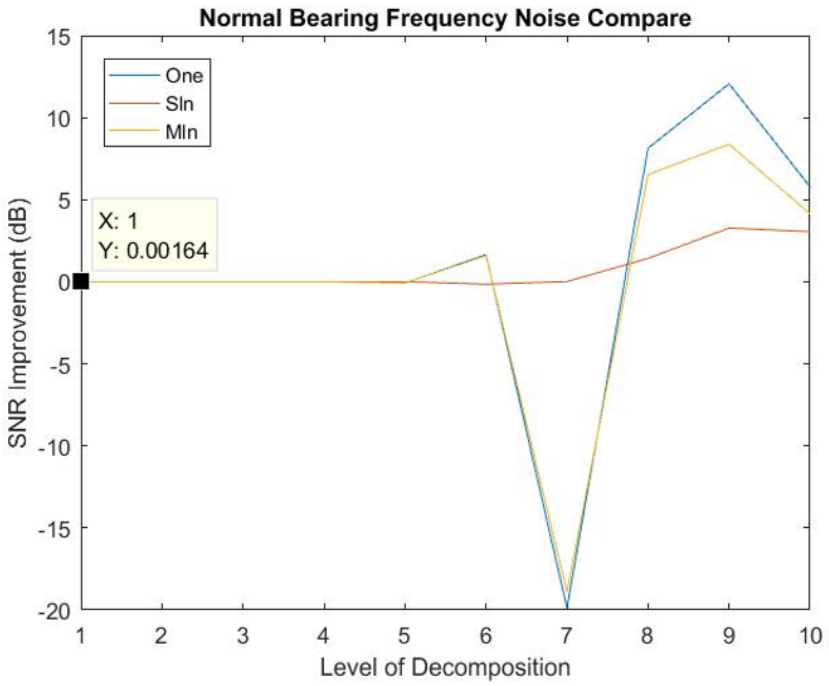

Figure 6.15: Short Time Signals - Fourier analysis, normal bearing noise scale estimation comparison.

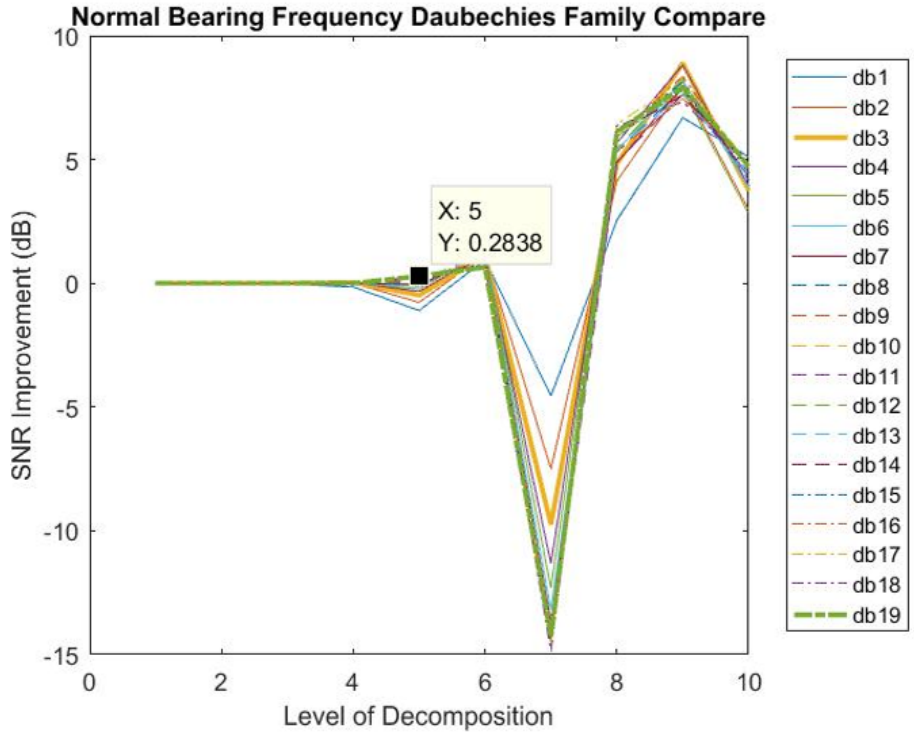

Figure 6.16: Short Time Signals - Fourier analysis, normal bearing Daubechies family comparison.

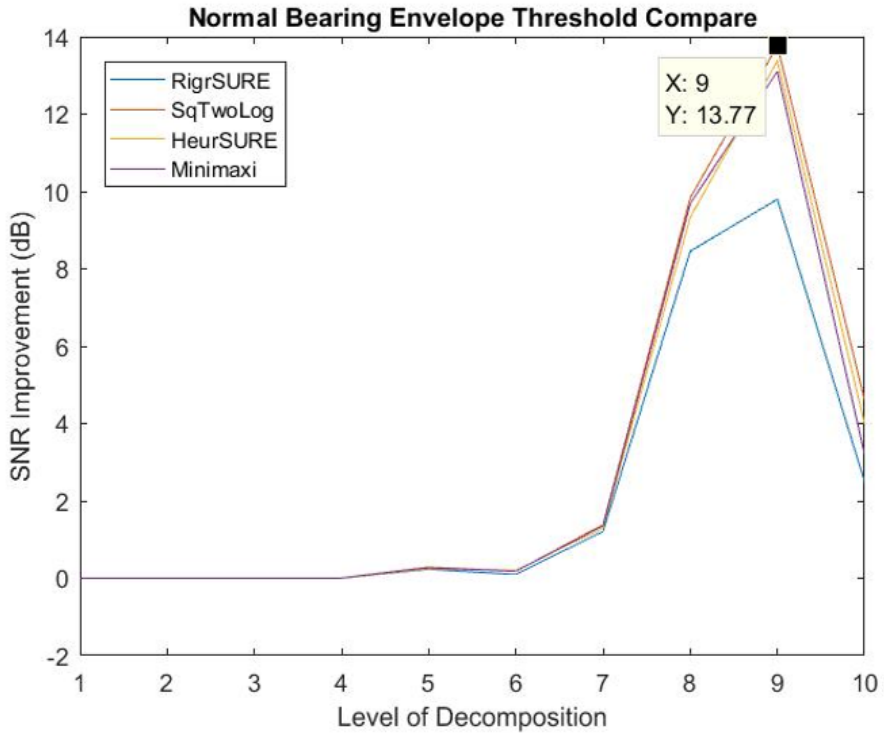

Figure 6.17: Short Time Signals - envelope analysis, normal bearing threshold comparison.

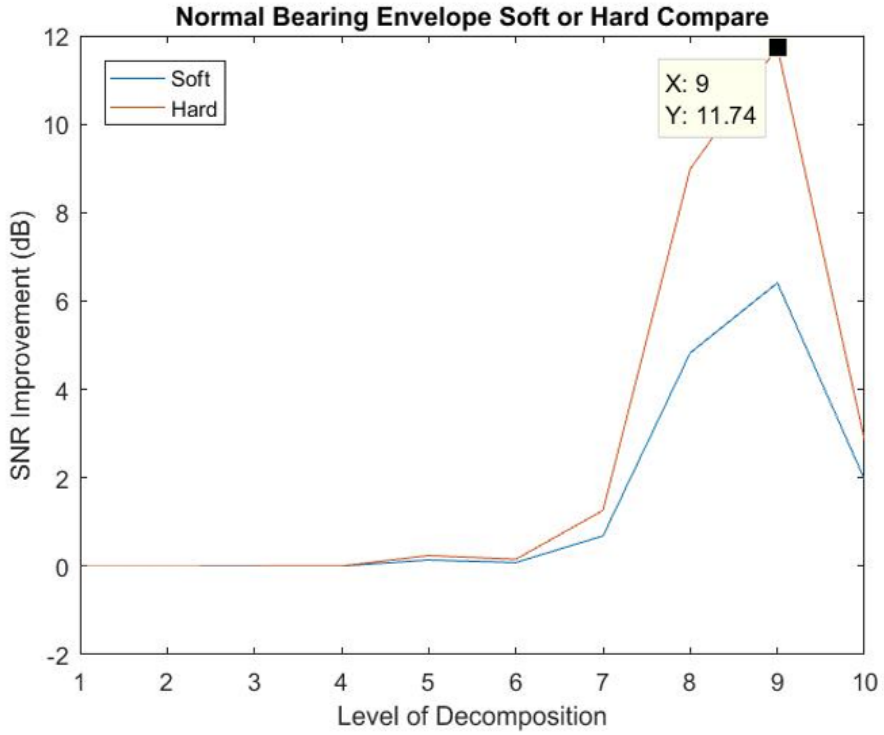

Figure 6.18: Short Time Signals - envelop analysis, normal bearing threshold application comparison.

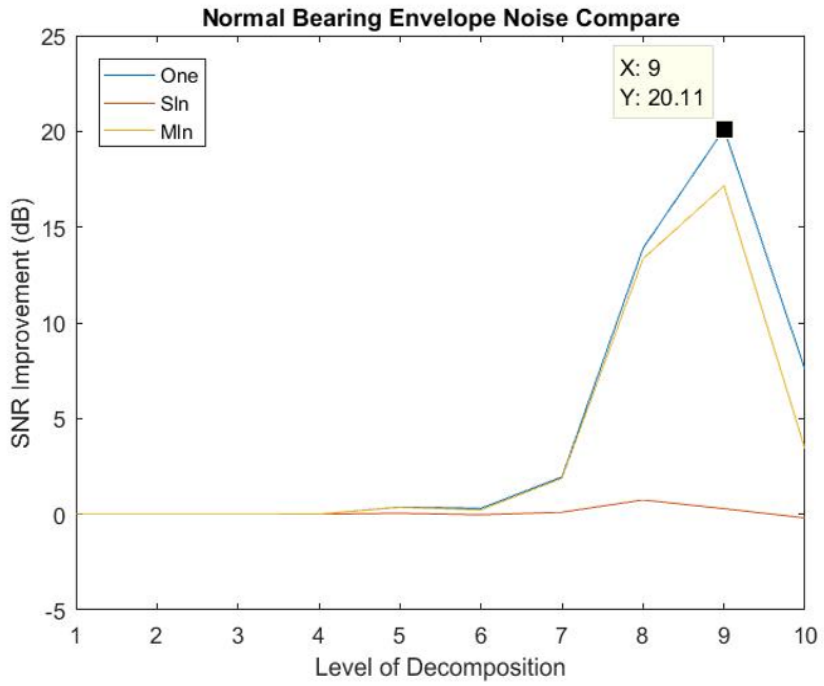

Figure 6.19: Short Time Signals - envelope analysis, normal bearing noise scale estimation comparison.

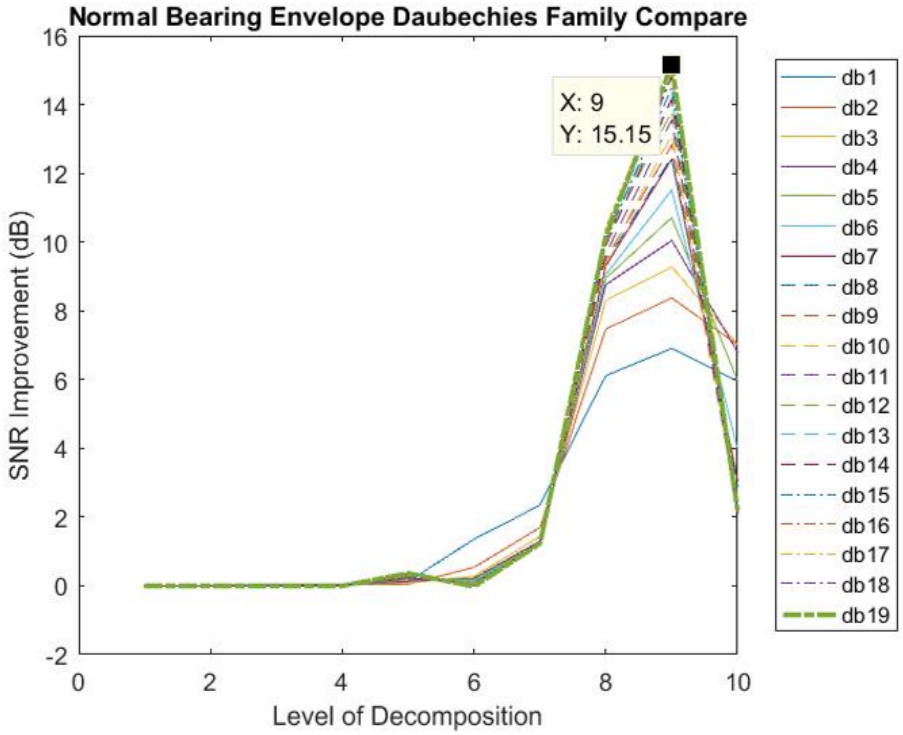

Figure 6.20: Short Time Signals - envelope analysis, normal bearing Daubechies family comparison.

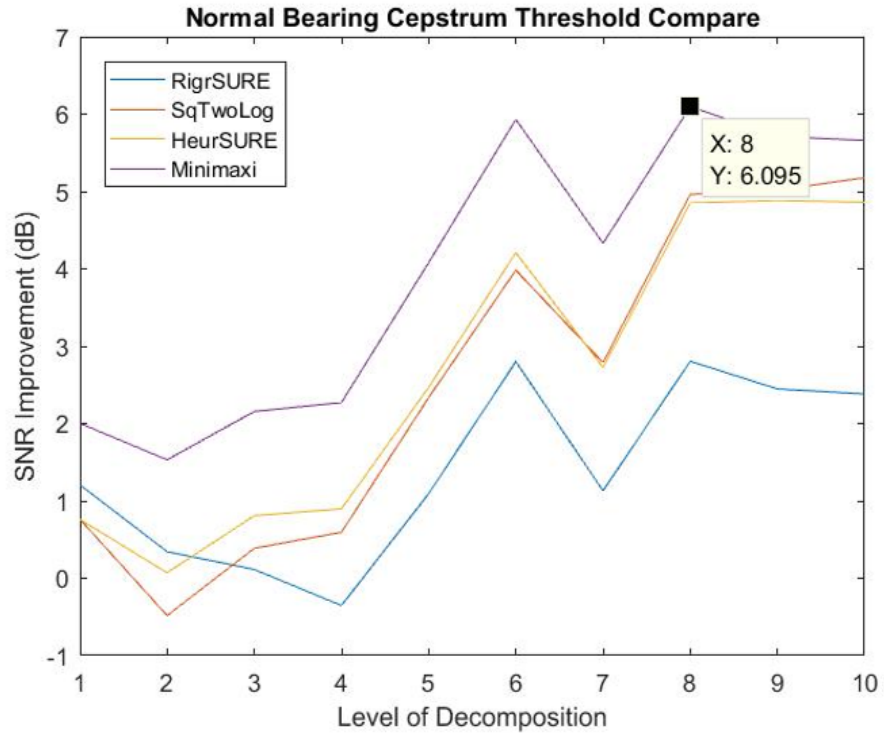

Figure 6.21: Short Time Signals - cepstrum analysis, normal bearing threshold comparison.

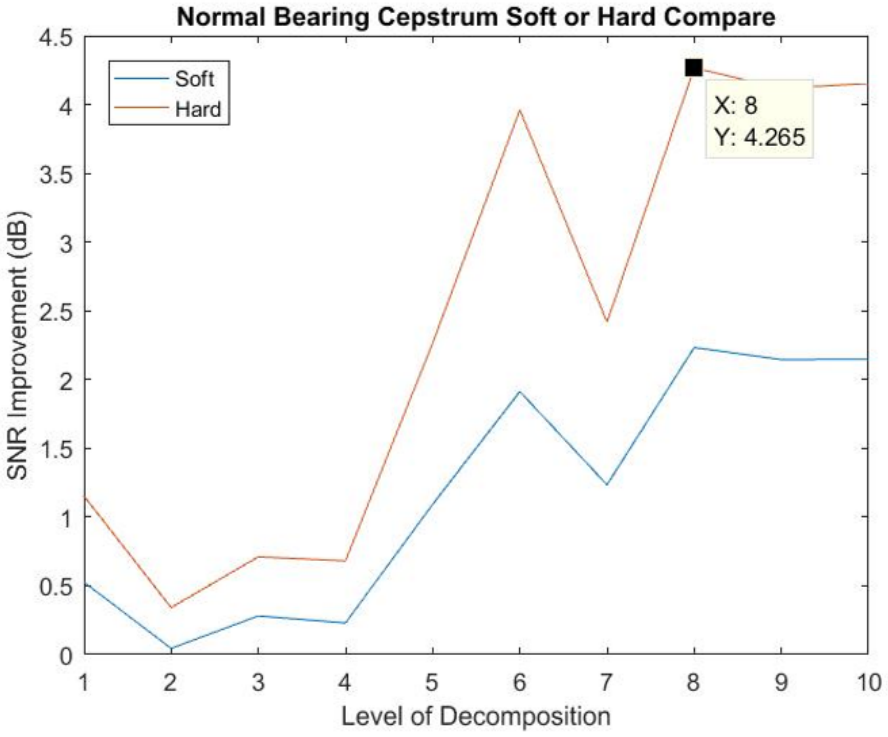

Figure 6.22: Short Time Signals - cepstrum analysis, normal bearing threshold application comparison.

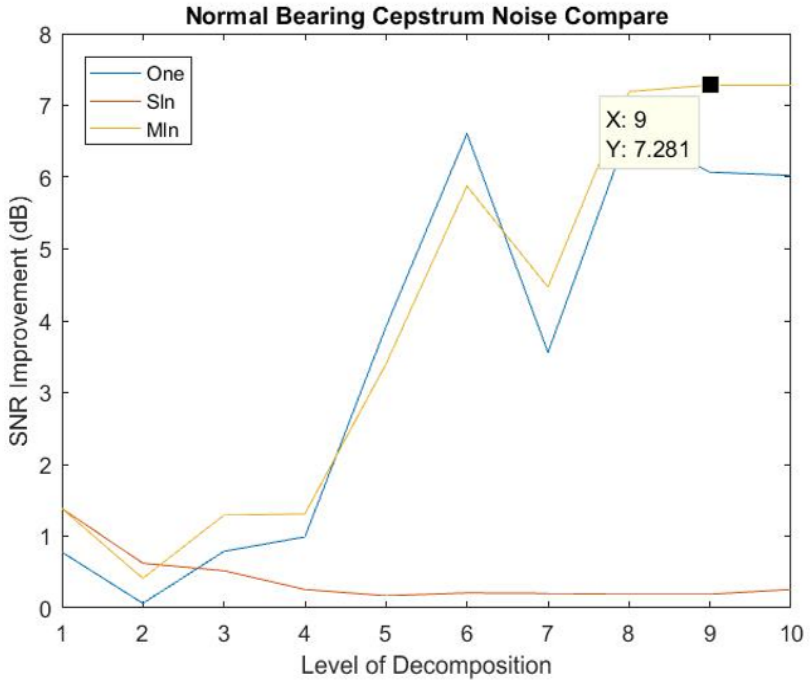

Figure 6.23: Short Time Signals - cepstrum analysis, normal bearing noise scale estimation comparison.

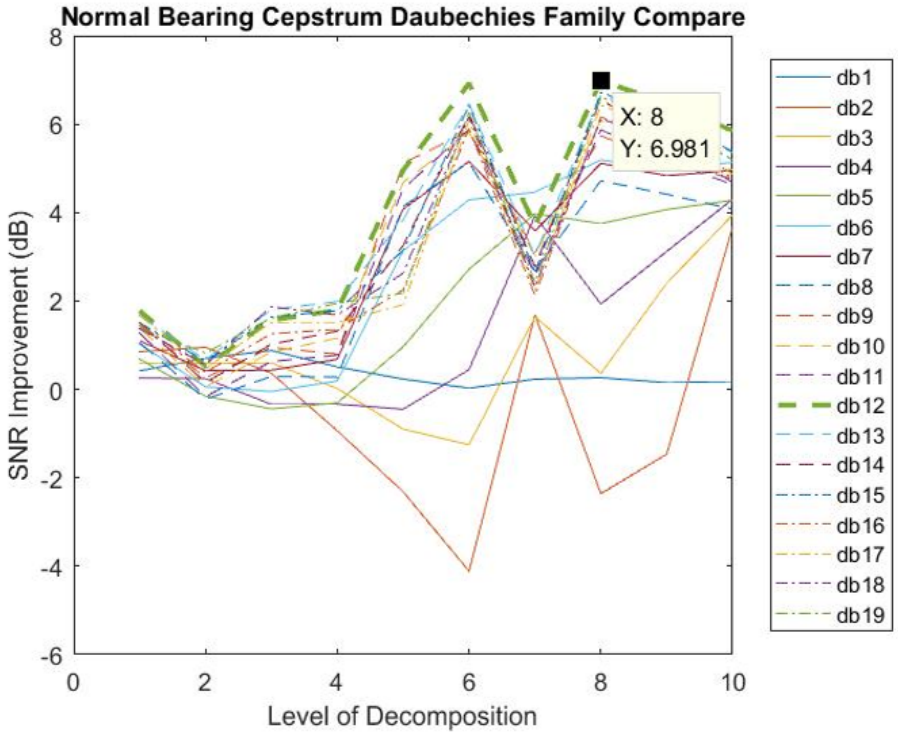

Figure 6.24: Short Time Signals - cepstrum analysis, normal bearing Daubechies family comparison.

6.3.3 Short Time Signals - Rolling Element Faults

Case Western Reserve Bearing Data Centre was the only source of bearing vibration signals with rolling element faults. Faults were artificially seeded in these bearings using electro-discharge machining. Four different magnitude of faults were seeded and the vibration signal for each fault was recorded at four different bearing loads.

The following graphs provide the average SNR improvement as a result of wavelet denoising for bearings with rolling element faults. 16 files were analysed and the results averaged. Figures 6.25, 6.26, 6.27 and 6.28 summarise the SNR improvement when Fourier analysis was used to extract the CFFs. Figures 6.29, 6.30, 6.31 and 6.32 summarise the SNR improvement when envelope analysis was used to extract the CFFs. Figures 6.33, 6.34, 6.35 and 6.36 summarise the SNR improvement when cepstrum analysis was used to extract the CFFs. On the graphs displaying the Daubechies family comparison the most effective Daubechies wavelet is boldly coloured.

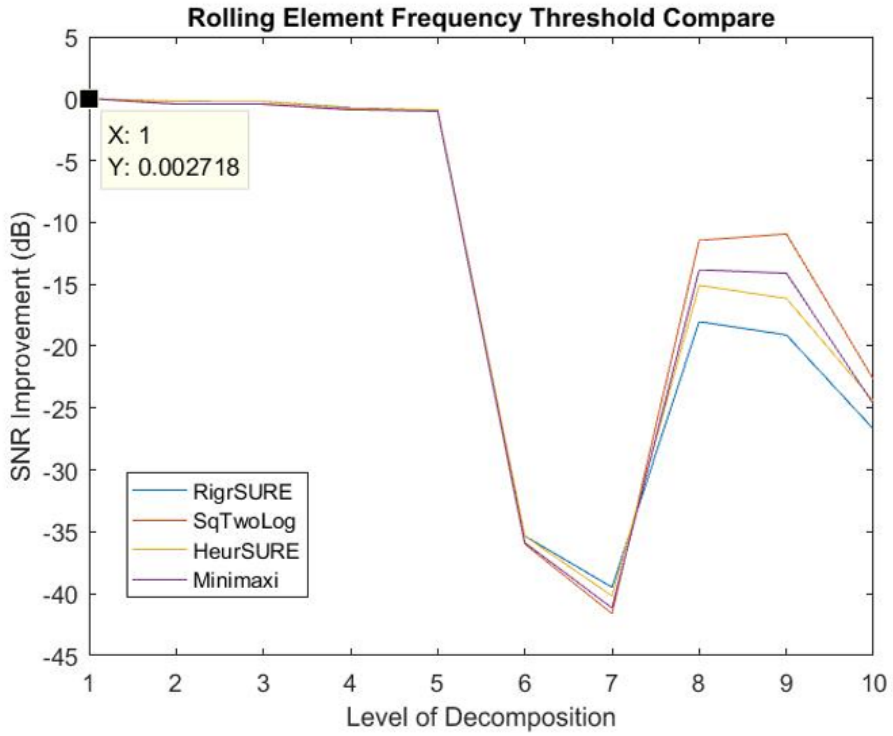

Figure 6.25: Short Time Signals - Fourier analysis, rolling element threshold comparison.

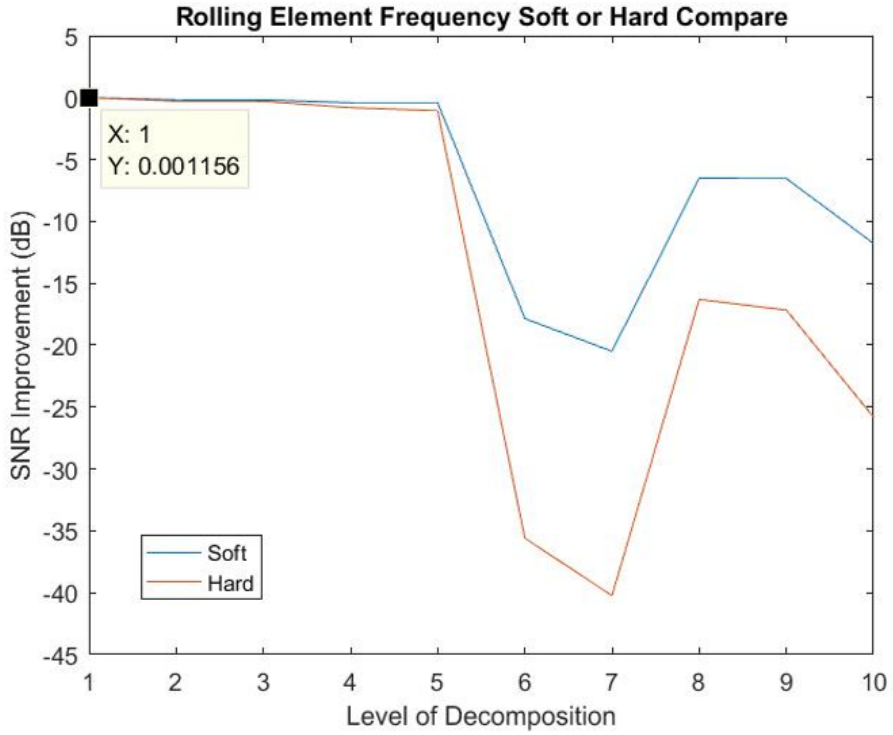

Figure 6.26: Short Time Signals - Fourier analysis, rolling element threshold application comparison.

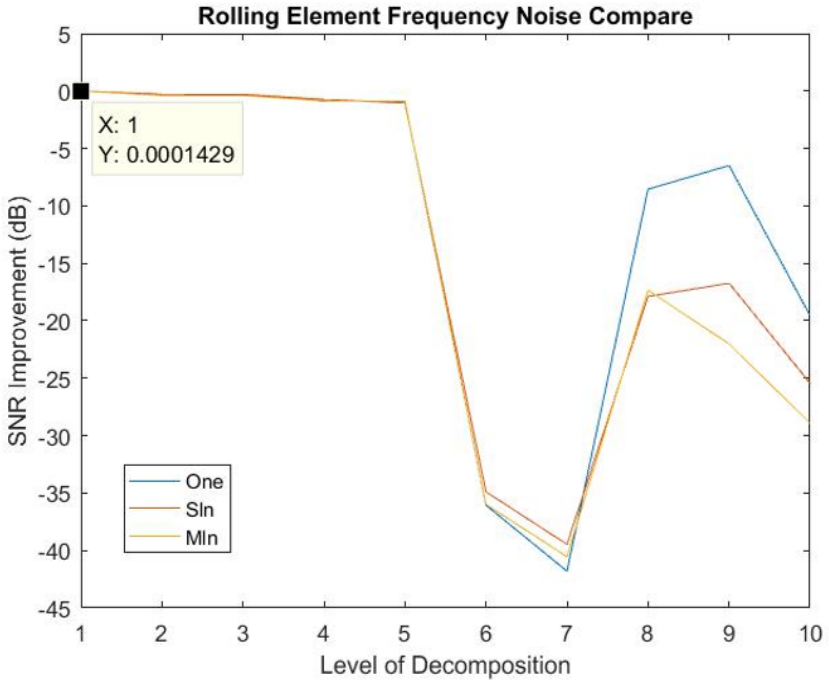

Figure 6.27: Short Time Signals - Fourier analysis, rolling element noise scale estimation comparison.

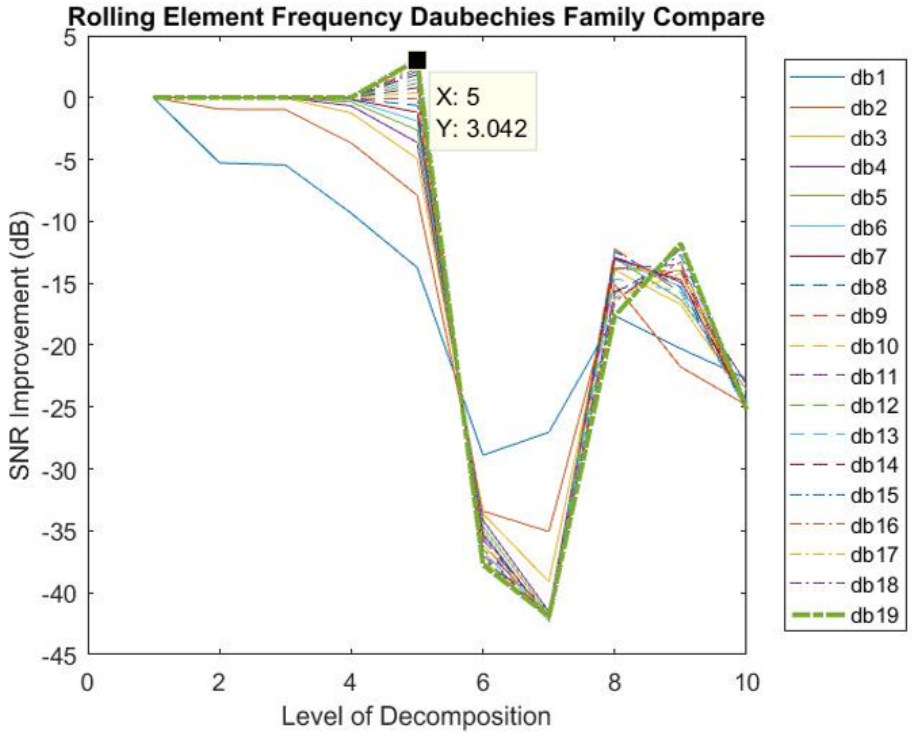

Figure 6.28: Short Time Signals - Fourier analysis, rolling element Daubechies family comparison.

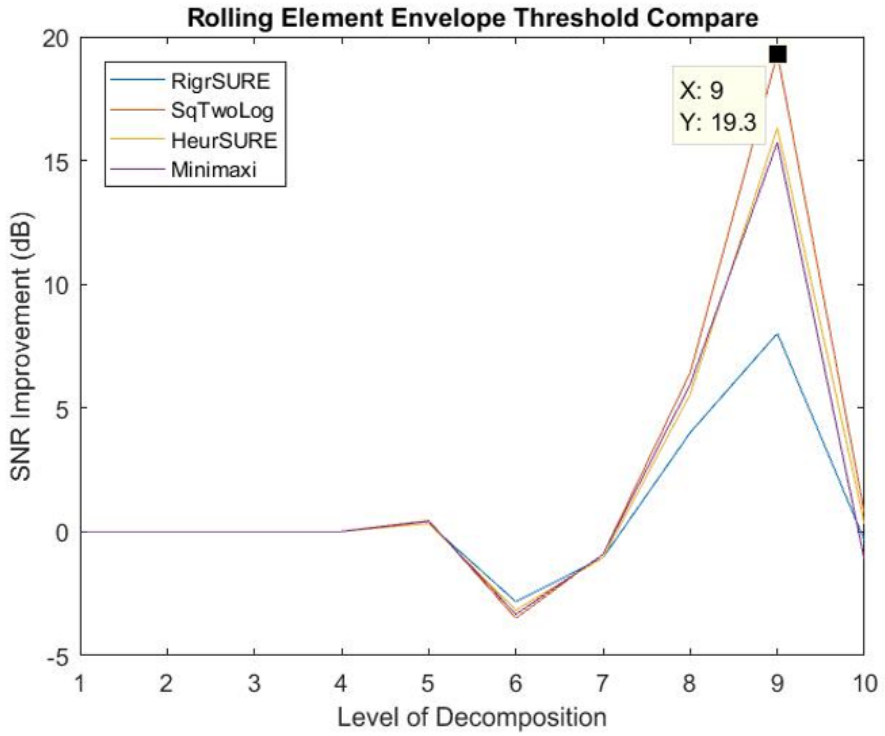

Figure 6.29: Short Time Signals - envelope analysis, rolling element threshold comparison.

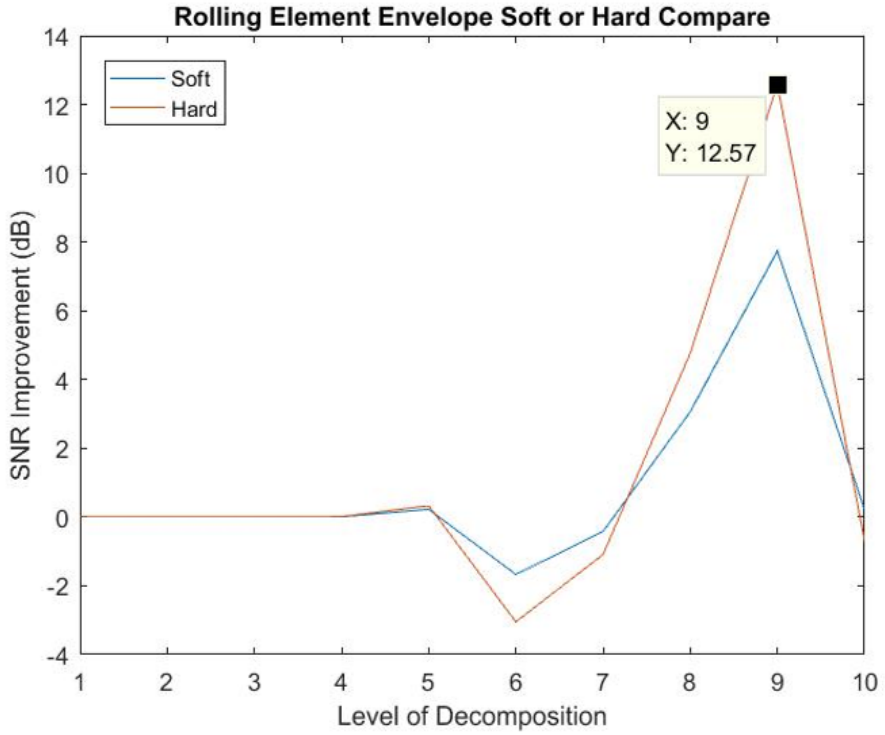

Figure 6.30: Short Time Signals - envelop analysis, rolling element threshold application comparison.

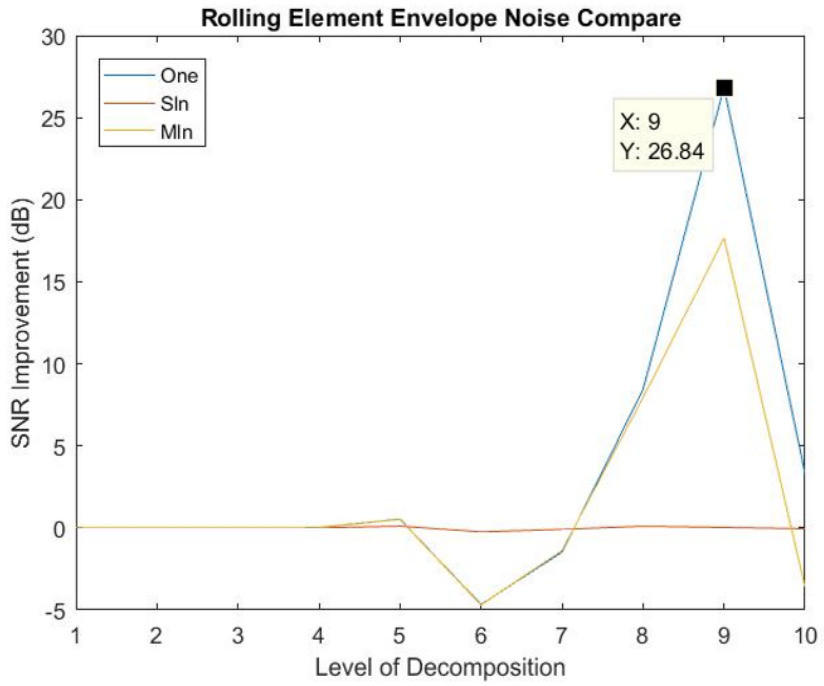

Figure 6.31: Short Time Signals - envelope analysis, rolling element noise scale estimation comparison.

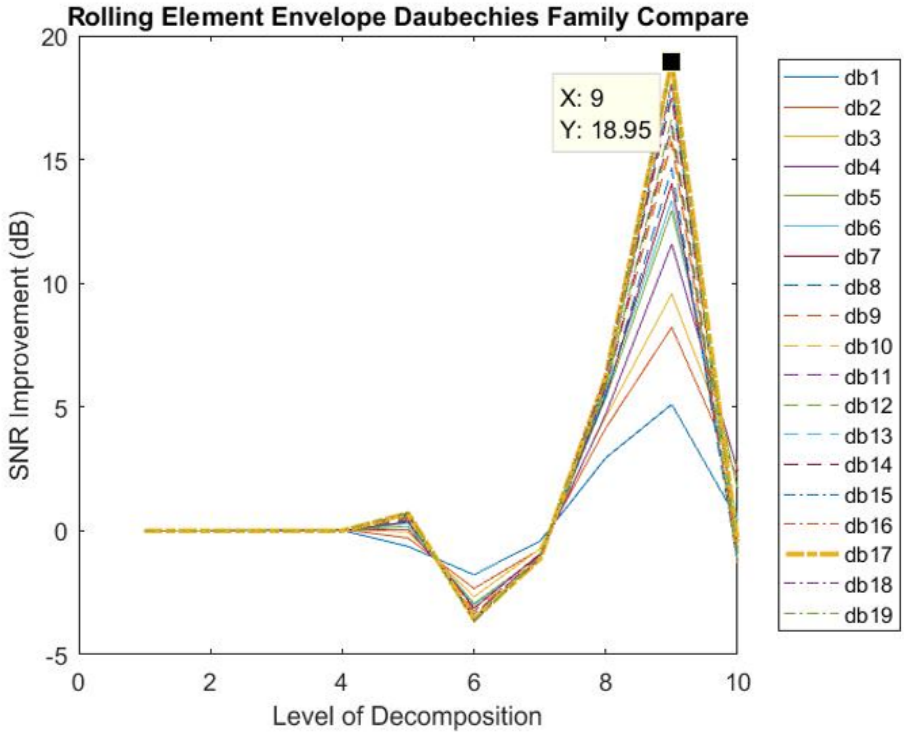

Figure 6.32: Short Time Signals - envelope analysis, rolling element Daubechies family comparison.

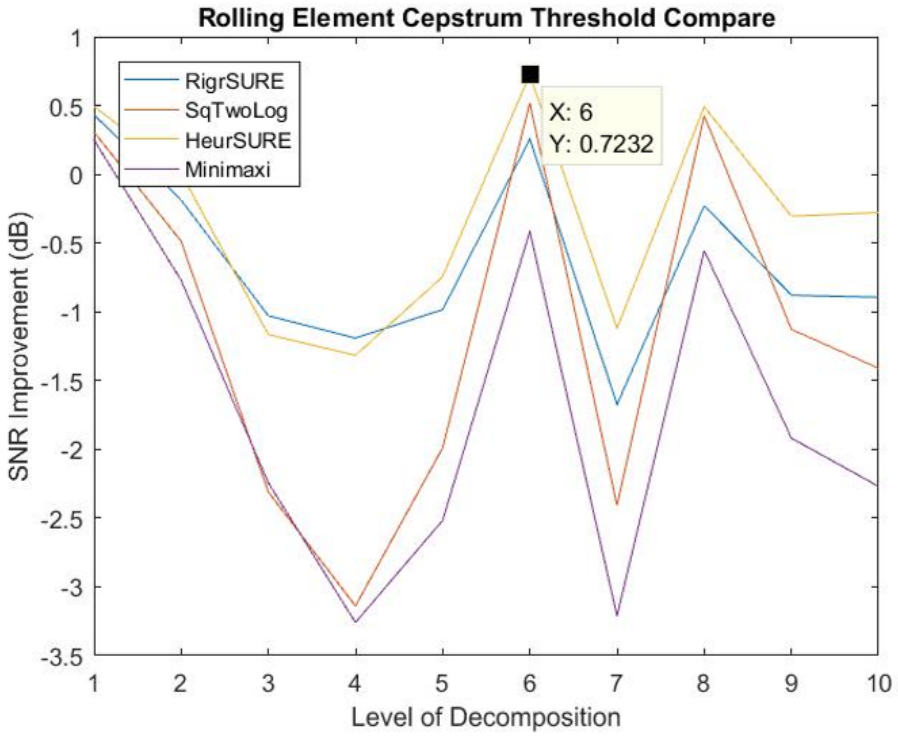

Figure 6.33: Short Time Signals - cepstrum analysis, rolling element threshold comparison.

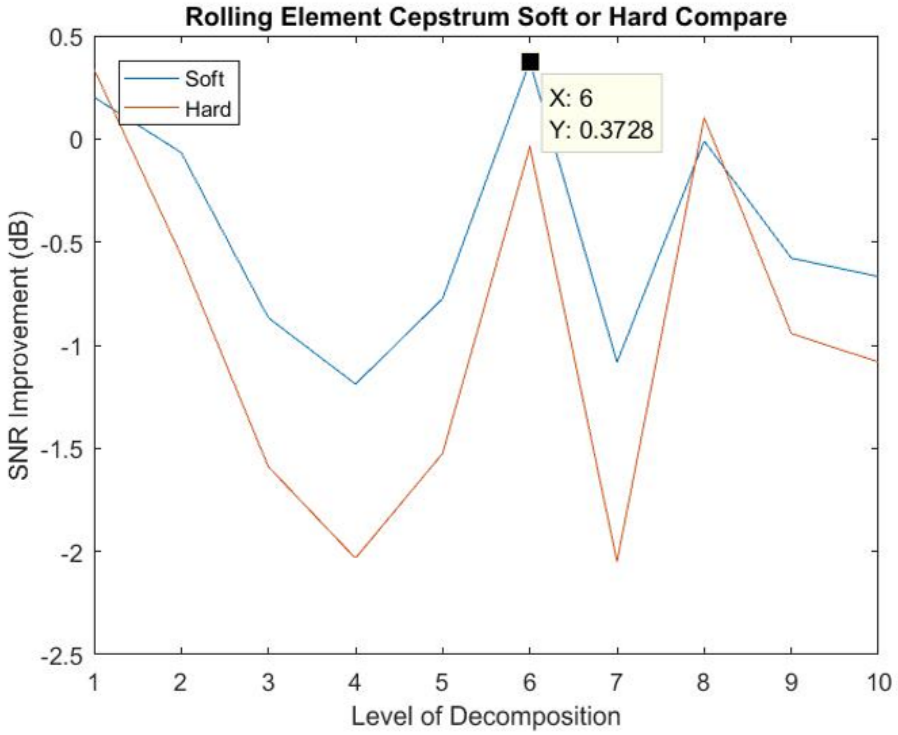

Figure 6.34: Short Time Signals - cepstrum analysis, rolling element threshold application comparison.

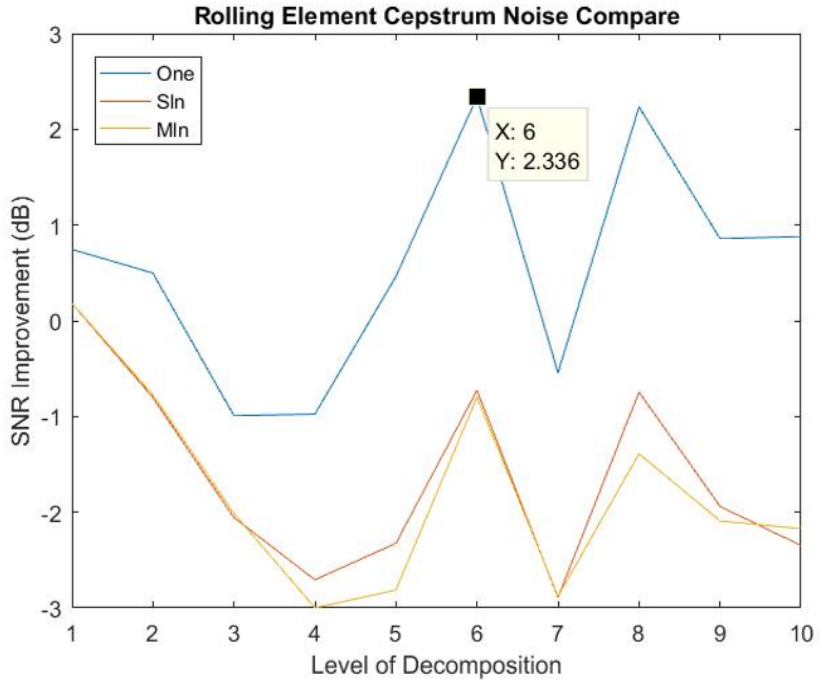

Figure 6.35: Short Time Signals - cepstrum analysis, rolling element noise scale estimation comparison.

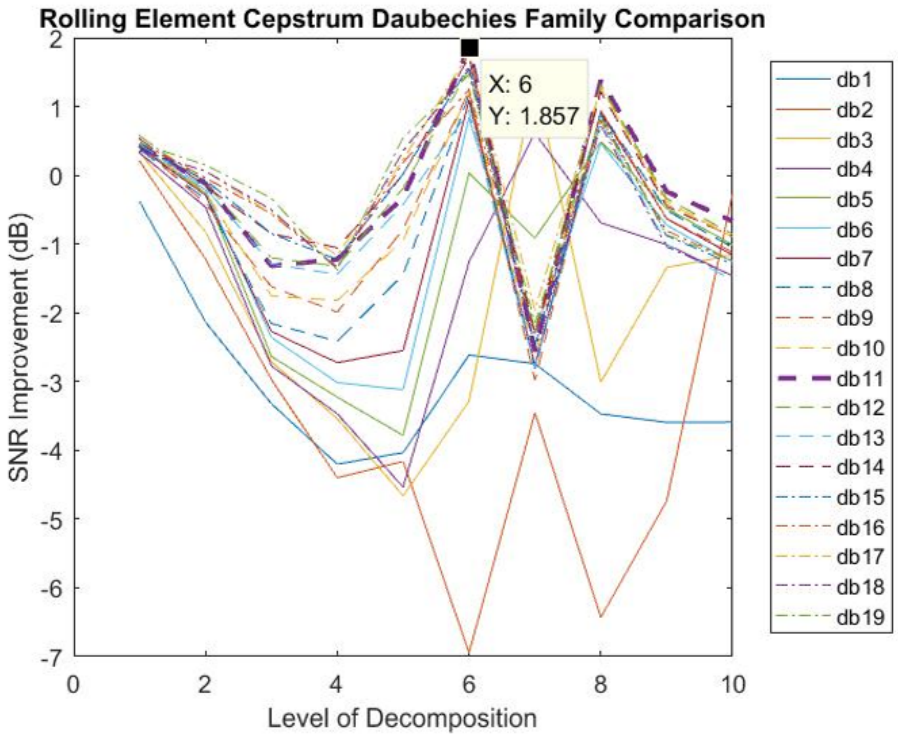

Figure 6.36: Short Time Signals - cepstrum analysis, rolling element Daubechies family comparison.

6.3.4 Short Time Signals - Inner Race Faults

Inner race fault bearing files were sourced from Case Western Reserve Bearing Data Centre, Machinery Failure Prevention Technology group and data-acoustic.com.

The following graphs provide the average SNR improvement as a result of wavelet denoising for bearings with no failure mode present. 24 files were analysed and the results averaged. Figures 6.37, 6.38, 6.39 and 6.40 summarise the SNR improvement when Fourier analysis was used to extract the CFFs. Figures 6.41, 6.42, 6.43 and 6.44 summarise the SNR improvement when envelope analysis was used to extract the CFFs. Figures 6.45, 6.46, 6.47 and 6.48 summarise the SNR improvement when cepstrum analysis was used to extract the CFFs. On the graphs displaying the Daubechies family comparison the most effective Daubechies wavelet is boldly coloured.

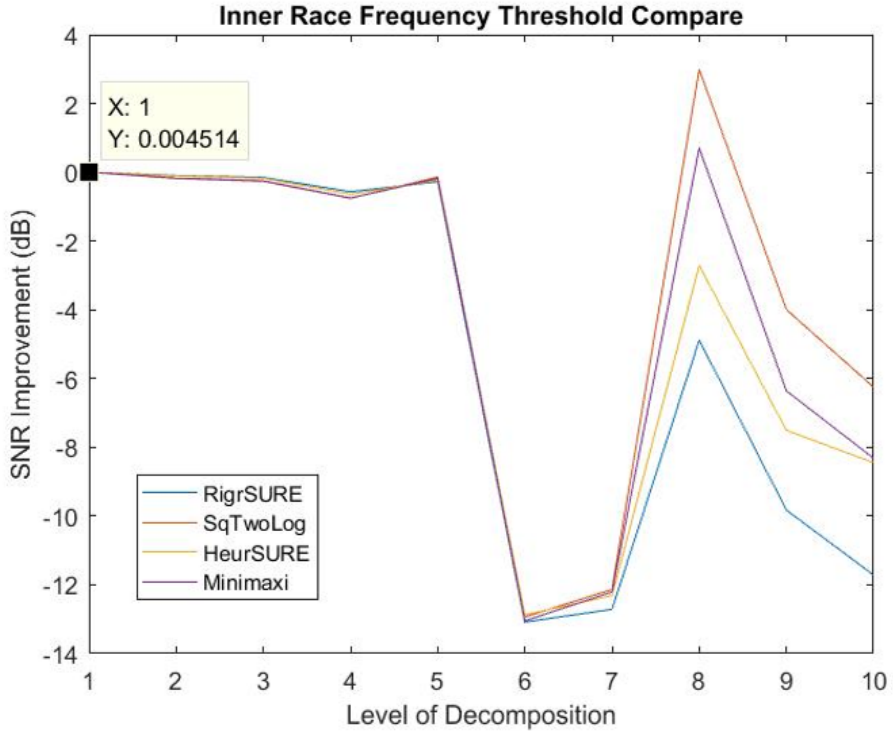

Figure 6.37: Short Time Signals - Fourier analysis, inner race threshold comparison.

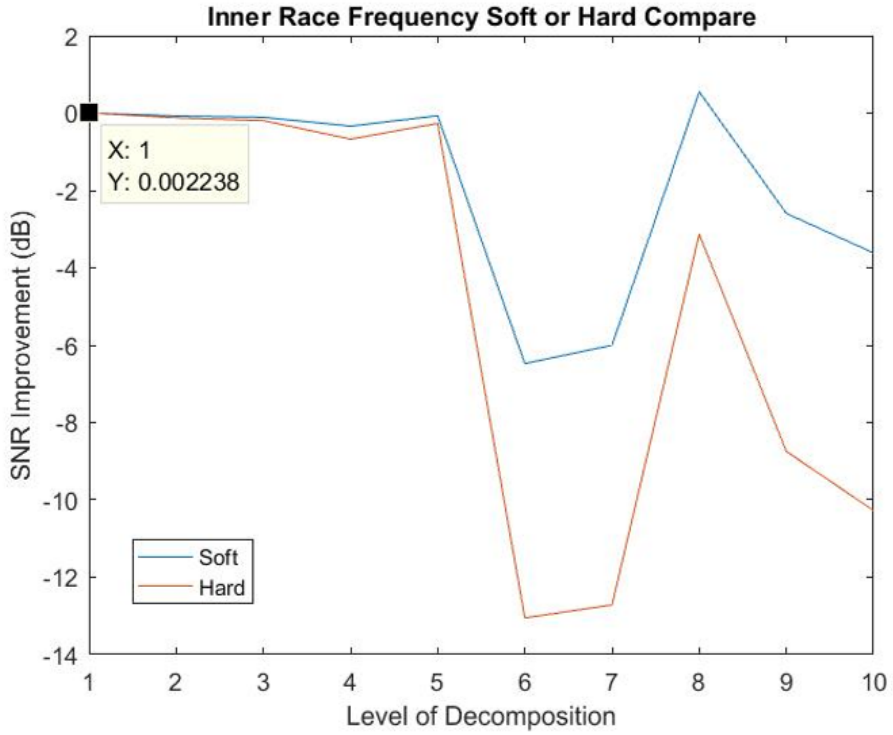

Figure 6.38: Short Time Signals - Fourier analysis, inner race threshold application comparison.

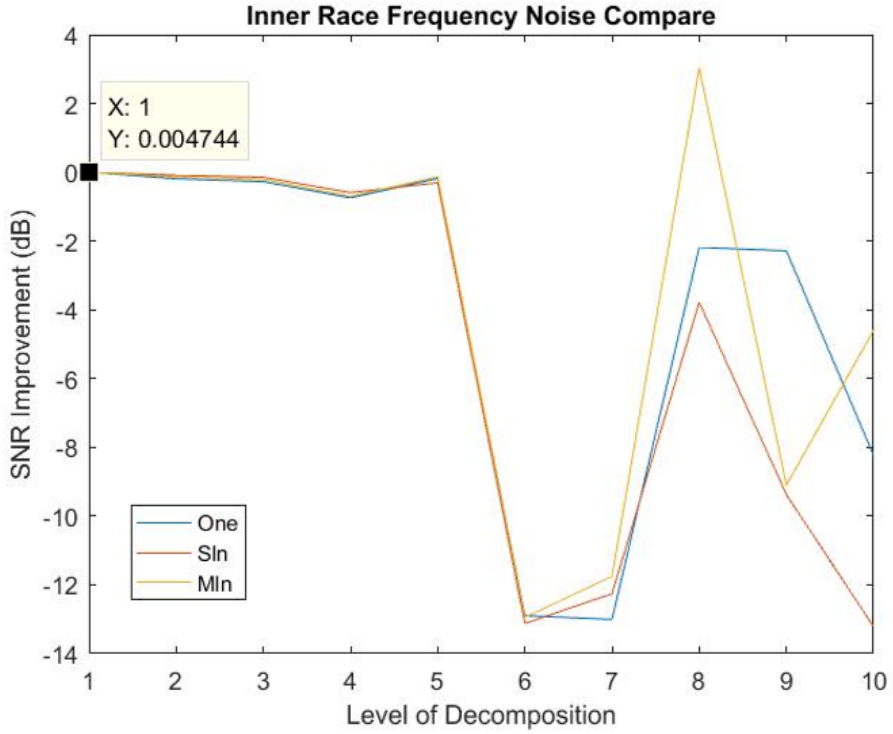

Figure 6.39: Short Time Signals - Fourier analysis, inner race noise scale estimation comparison.

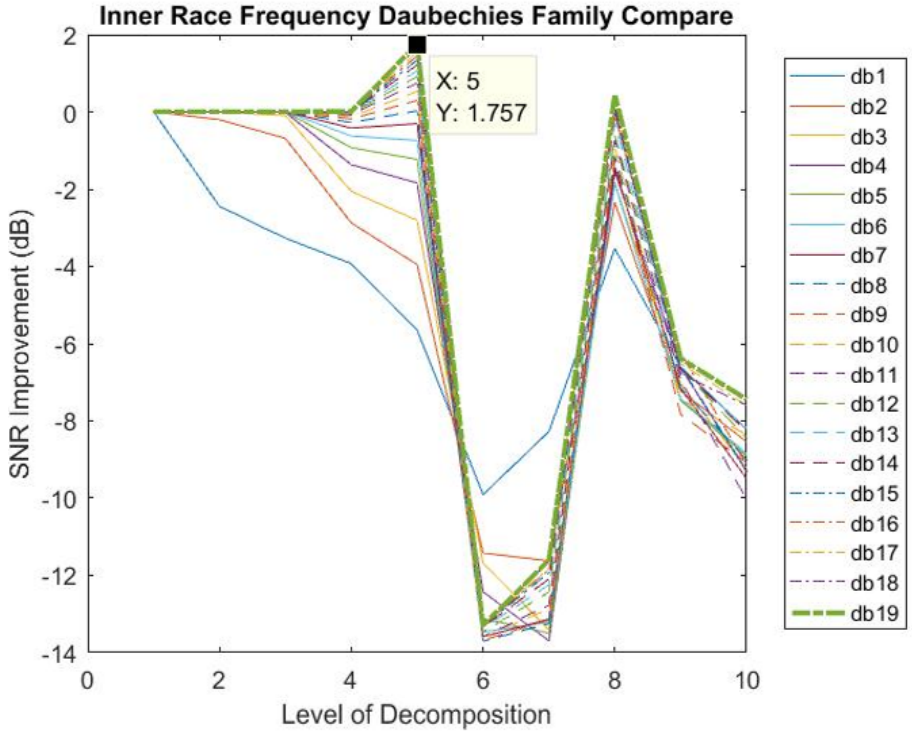

Figure 6.40: Short Time Signals - Fourier analysis, inner race Daubechies family comparison.

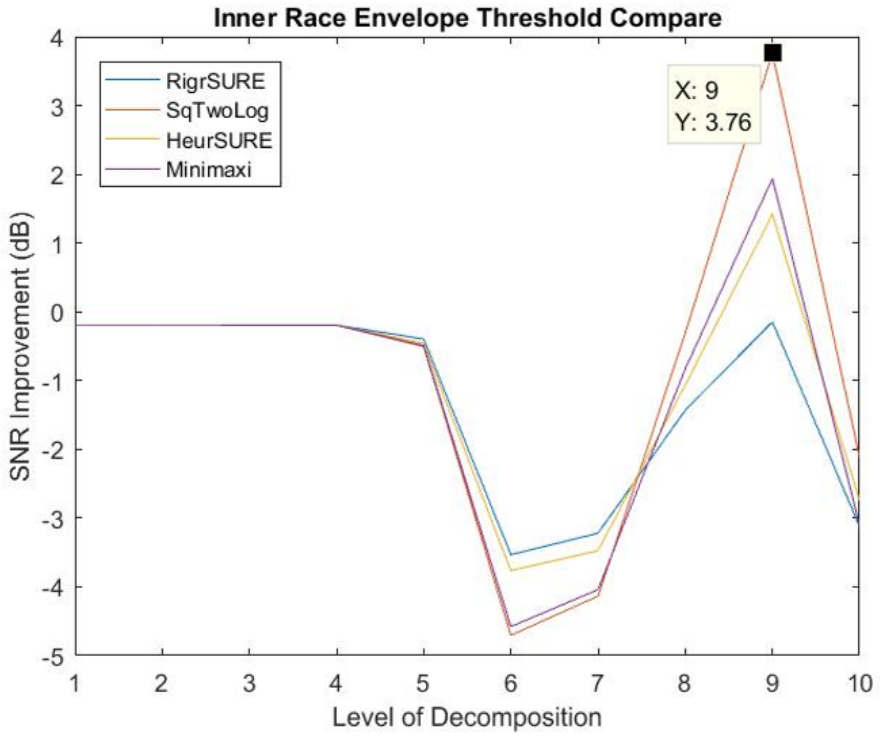

Figure 6.41: Short Time Signals - envelope analysis, inner race threshold comparison.

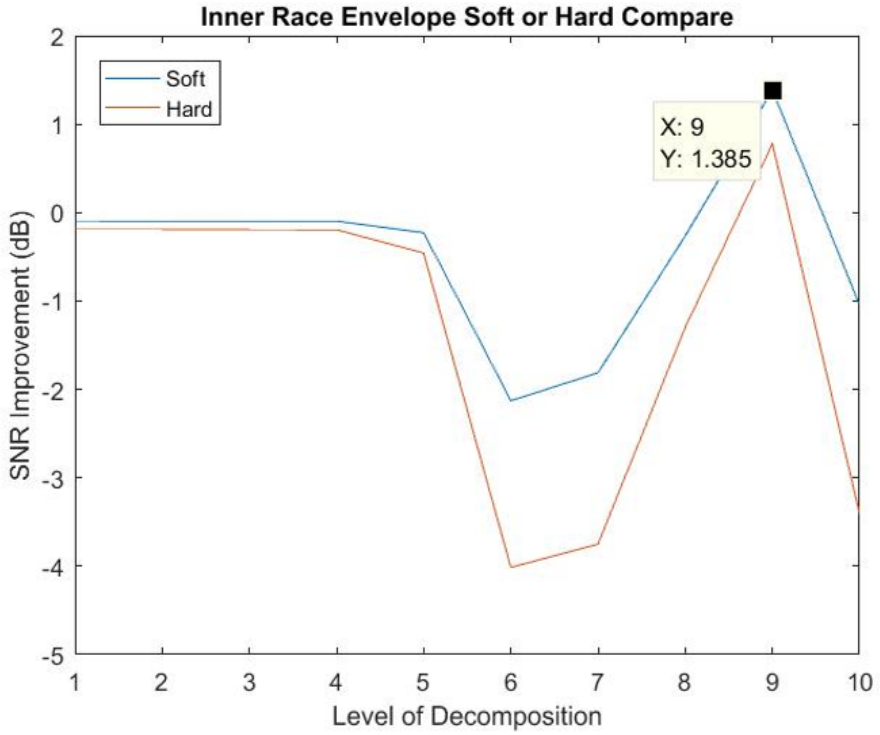

Figure 6.42: Short Time Signals - envelop analysis, inner race threshold application comparison.

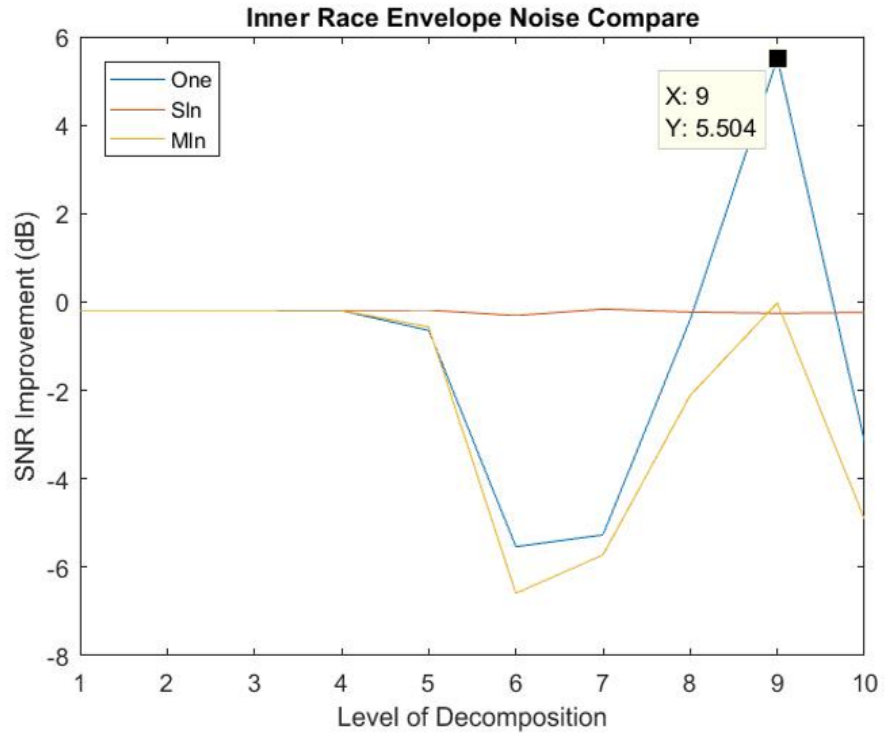

Figure 6.43: Short Time Signals - envelope analysis, inner race noise scale estimation comparison.

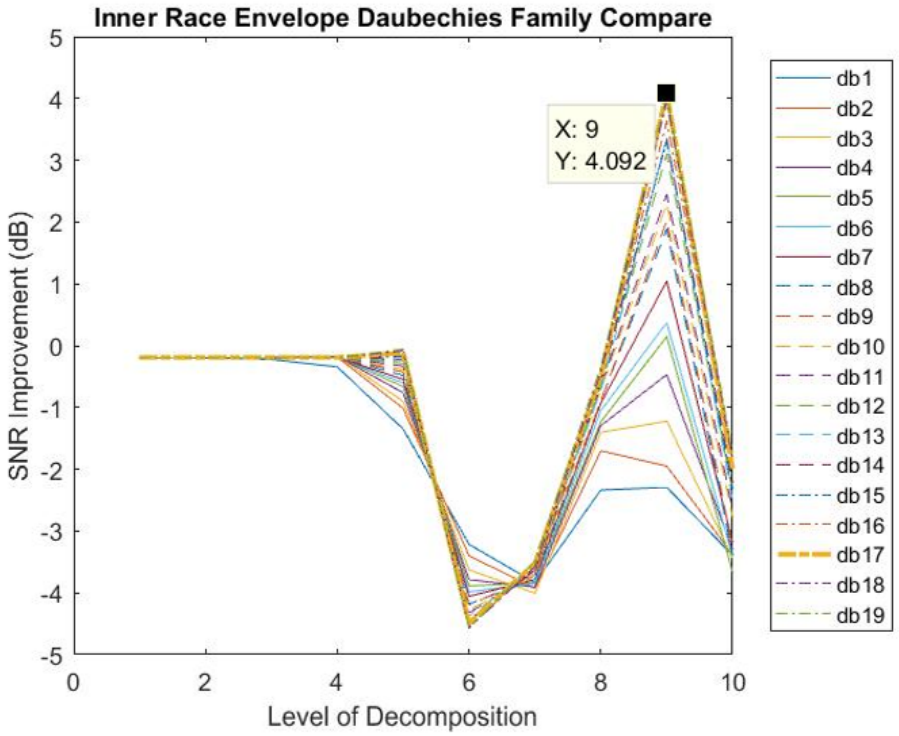

Figure 6.44: Short Time Signals - envelope analysis, inner race Daubechies family comparison.

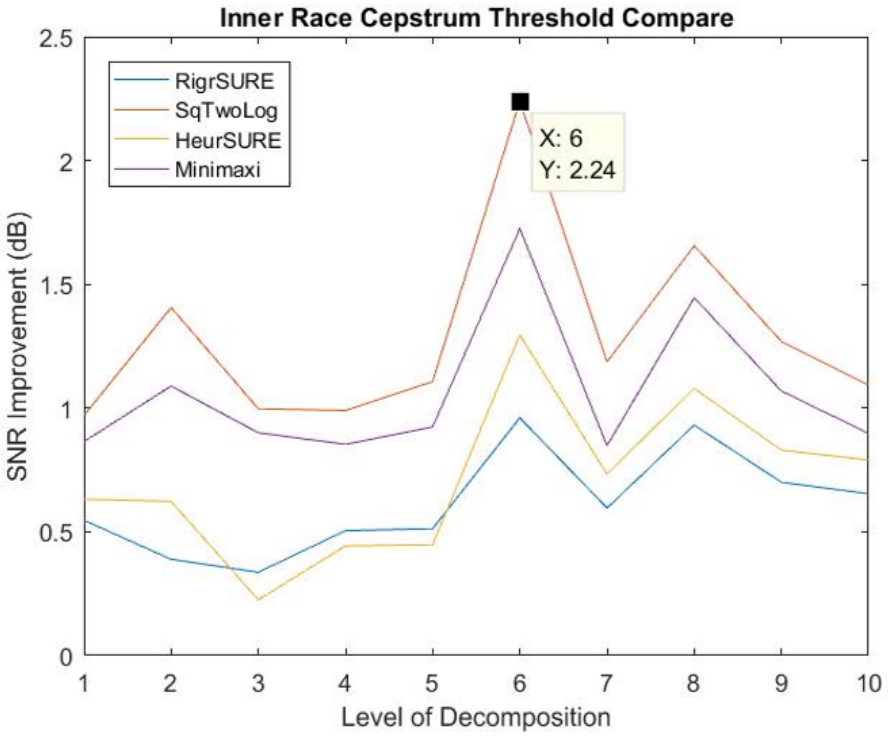

Figure 6.45: Short Time Signals - cepstrum analysis, inner race threshold comparison.

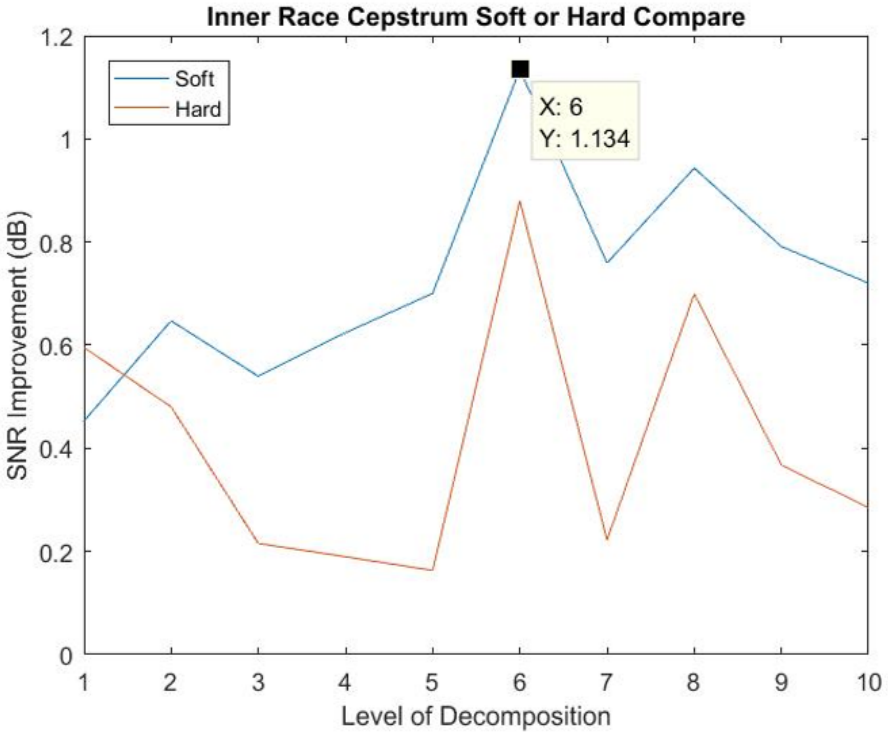

Figure 6.46: Short Time Signals - cepstrum analysis, inner race threshold application comparison.

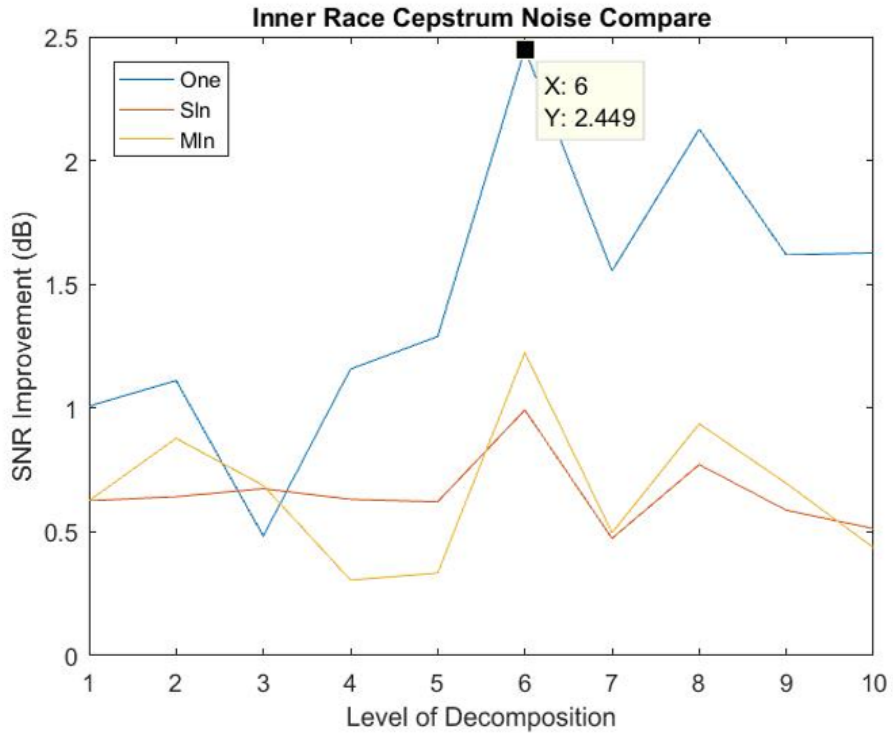

Figure 6.47: Short Time Signals - cepstrum analysis, inner race noise scale estimation comparison.

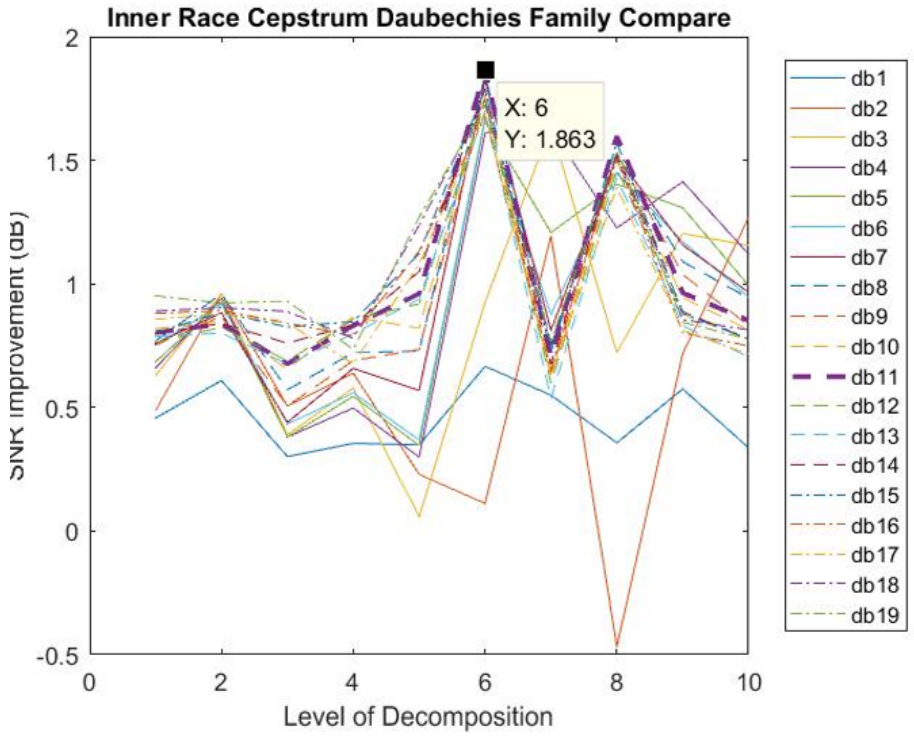

Figure 6.48: Short Time Signals - cepstrum analysis, inner race Daubechies family comparison.

6.3.5 Short Time Signals - Outer Race Faults

Outer race fault bearing files were sourced from Case Western Reserve Bearing Data Centre, Machinery Failure Prevention Technology group and data-acoustic.com.

The following graphs provide the average SNR improvement as a result of wavelet denoising for bearings with no failure mode present. 23 files were analysed and the results averaged. Figures 6.49, 6.50, 6.51 and 6.52 summarise the SNR improvement when Fourier analysis was used to extract the CFFs. Figures 6.53, 6.54, 6.55 and 6.56 summarise the SNR improvement when envelope analysis was used to extract the CFFs. Figures 6.57, 6.58, 6.59 and 6.60 summarise the SNR improvement when cepstrum analysis was used to extract the CFFs. On the graphs displaying the Daubechies family comparison the most effective Daubechies wavelet is boldly coloured.

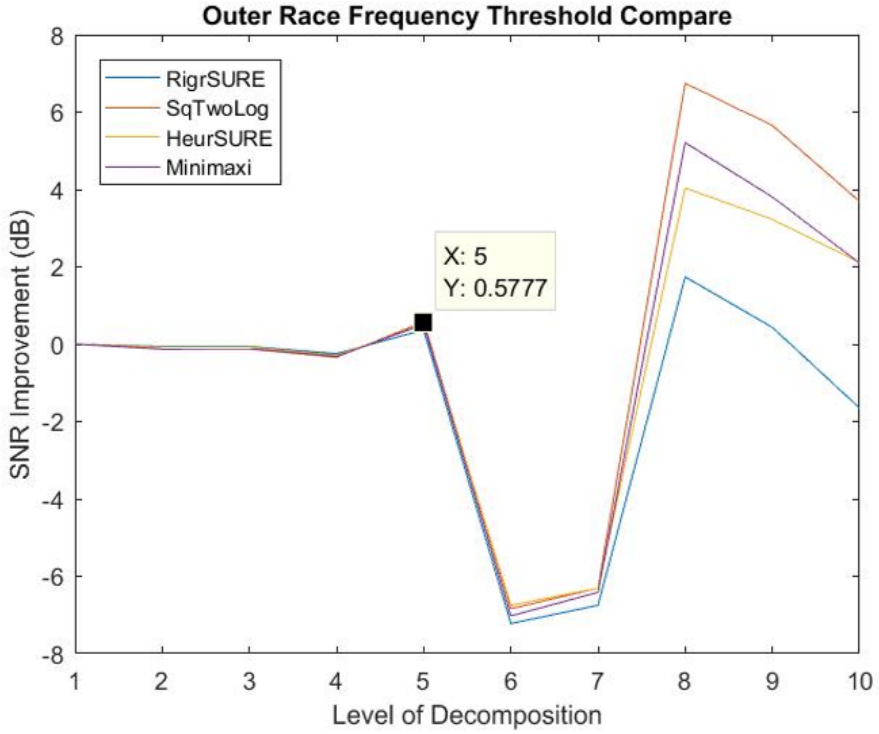

Figure 6.49: Short Time Signals - Fourier analysis, outer race threshold comparison.

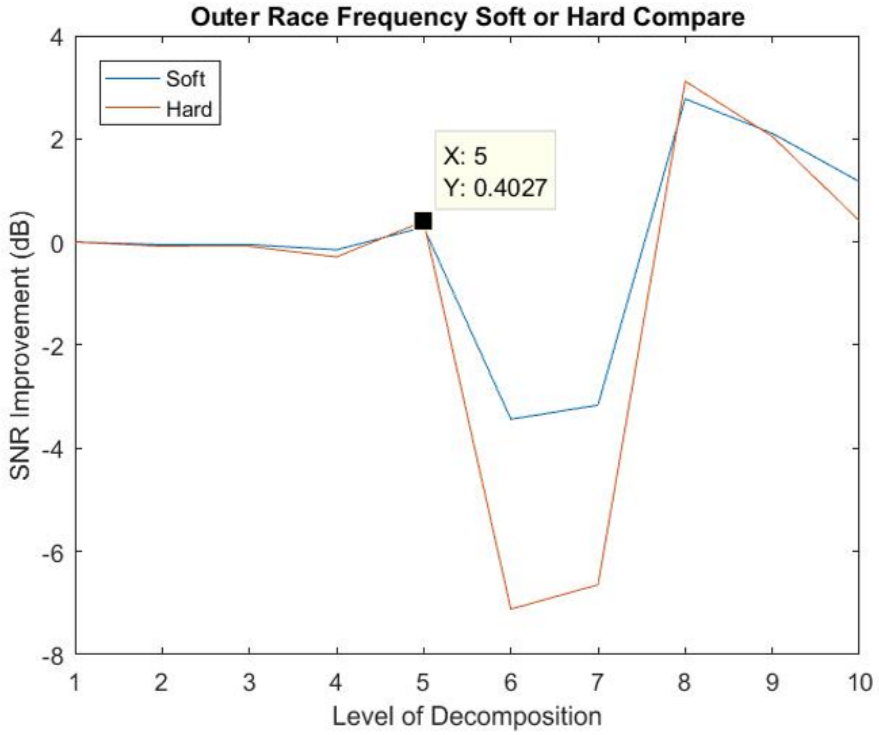

Figure 6.50: Short Time Signals - Fourier analysis, outer race threshold application comparison.

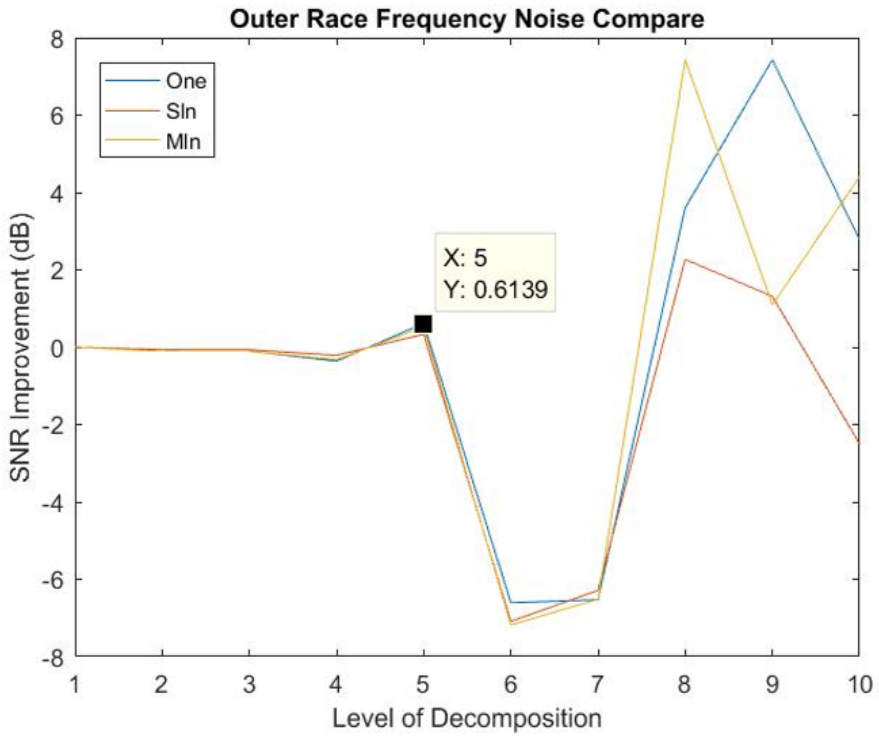

Figure 6.51: Short Time Signals - Fourier analysis, outer race noise estimation comparison.

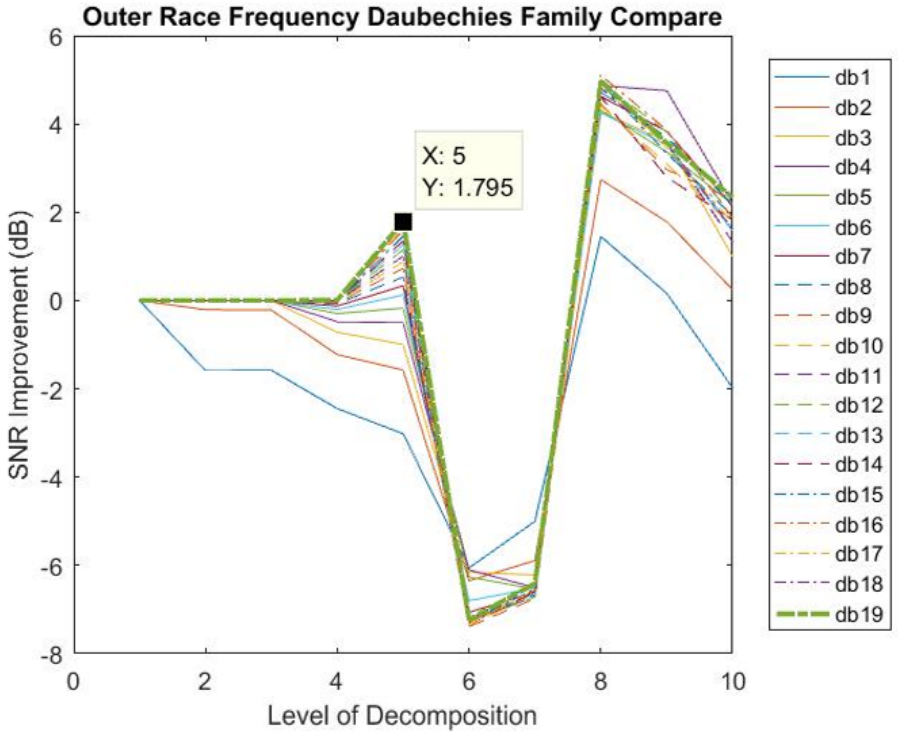

Figure 6.52: Short Time Signals - Fourier analysis, outer race Daubechies family comparison.

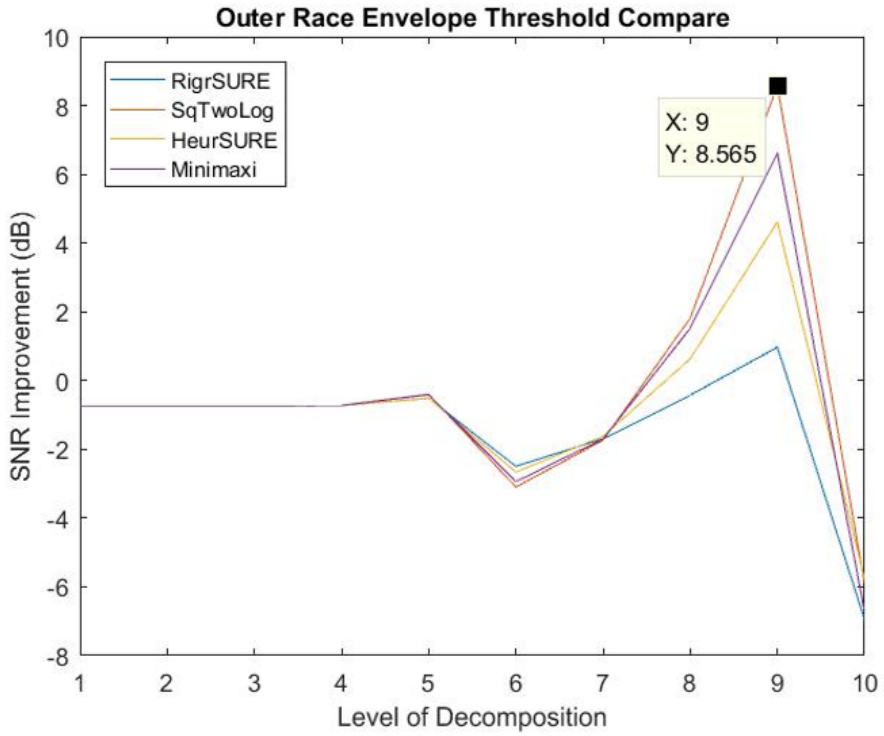

Figure 6.53: Short Time Signals - envelope analysis, outer race threshold comparison.

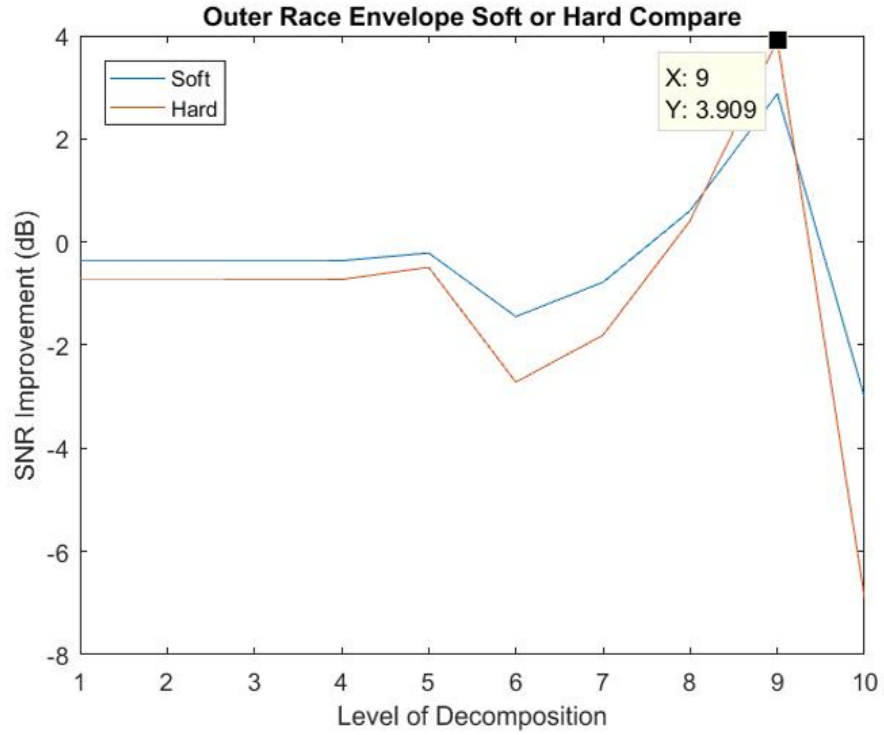

Figure 6.54: Short Time Signals - envelop analysis, outer race threshold application comparison.

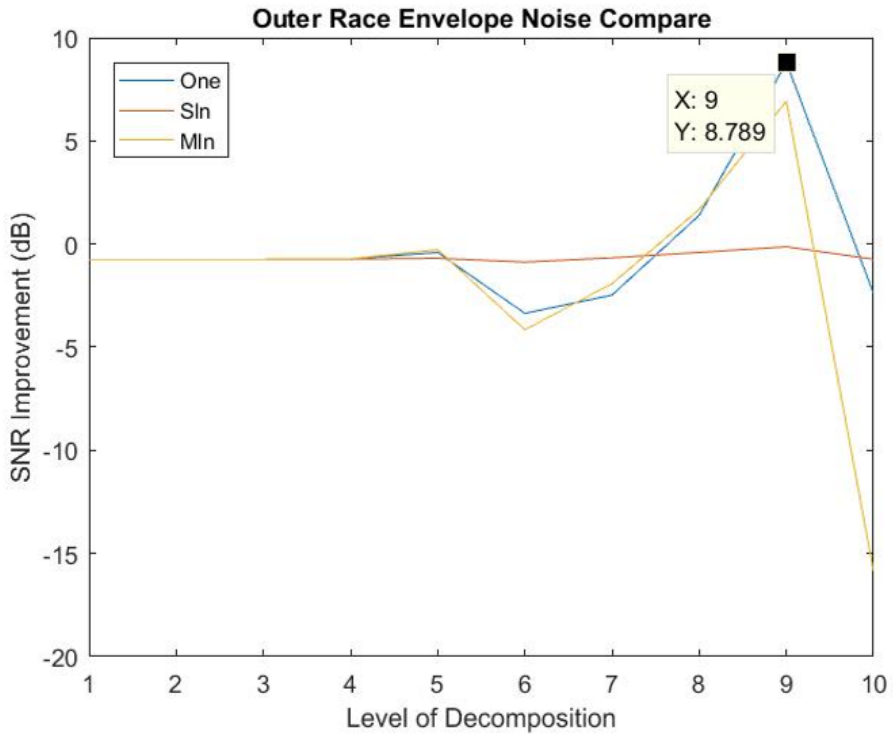

Figure 6.55: Short Time Signals - envelope analysis, outer race noise estimation comparison.

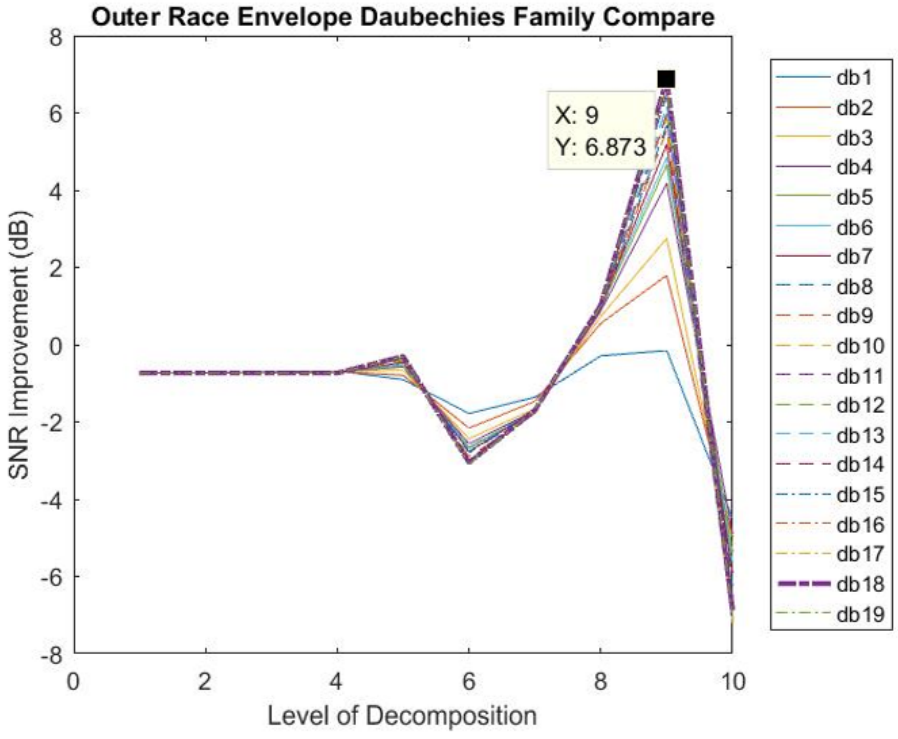

Figure 6.56: Short Time Signals - envelope analysis, outer race Daubechies family comparison.

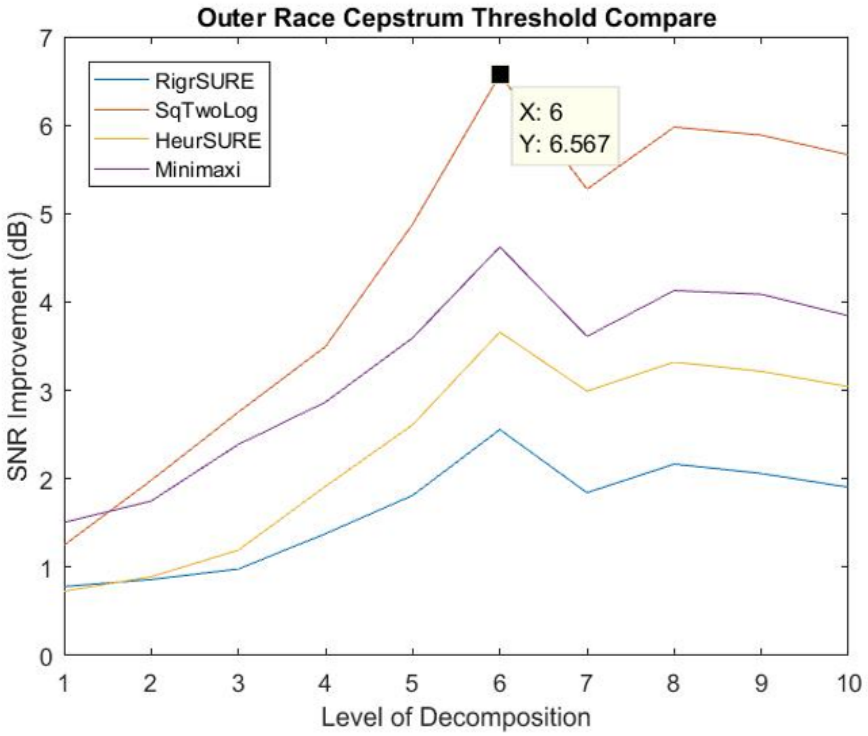

Figure 6.57: Short Time Signals - cepstrum analysis, outer race threshold comparison.

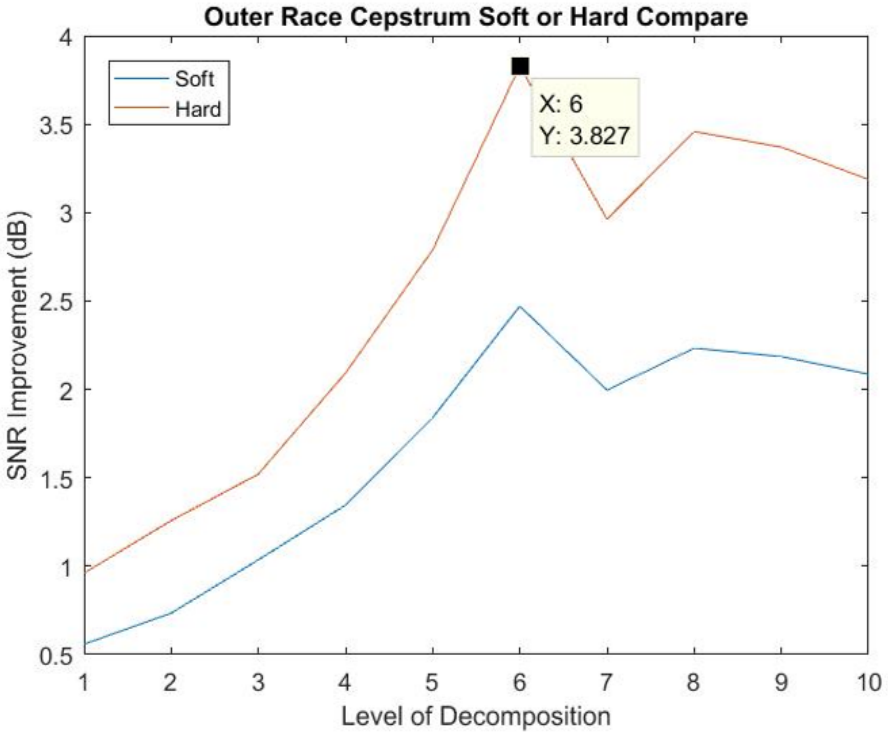

Figure 6.58: Short Time Signals - cepstrum analysis, outer race threshold application comparison.

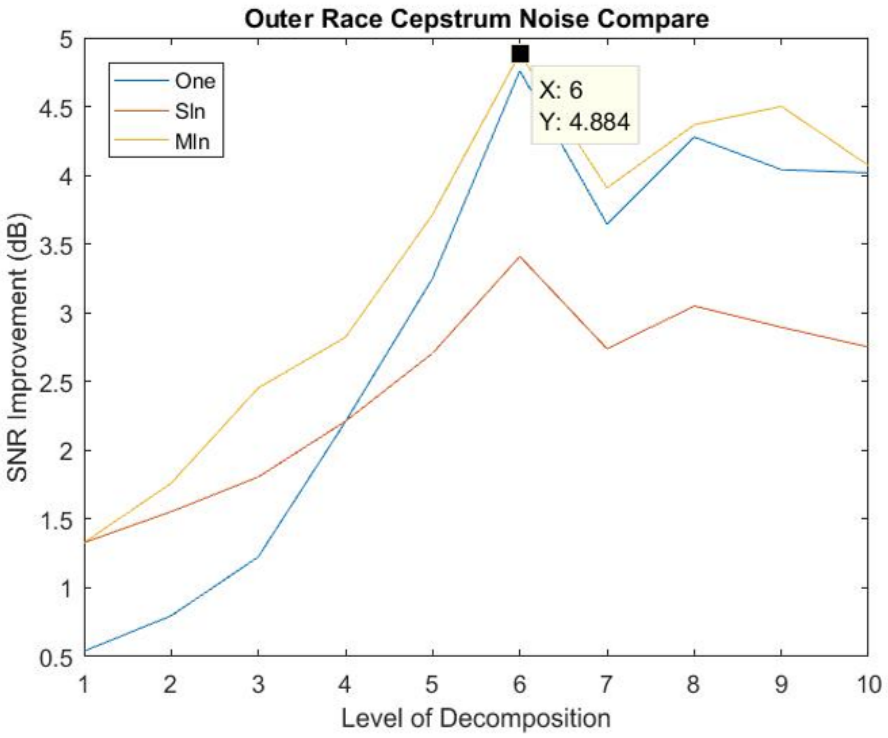

Figure 6.59: Short Time Signals - cepstrum analysis, outer race noise estimation comparison.

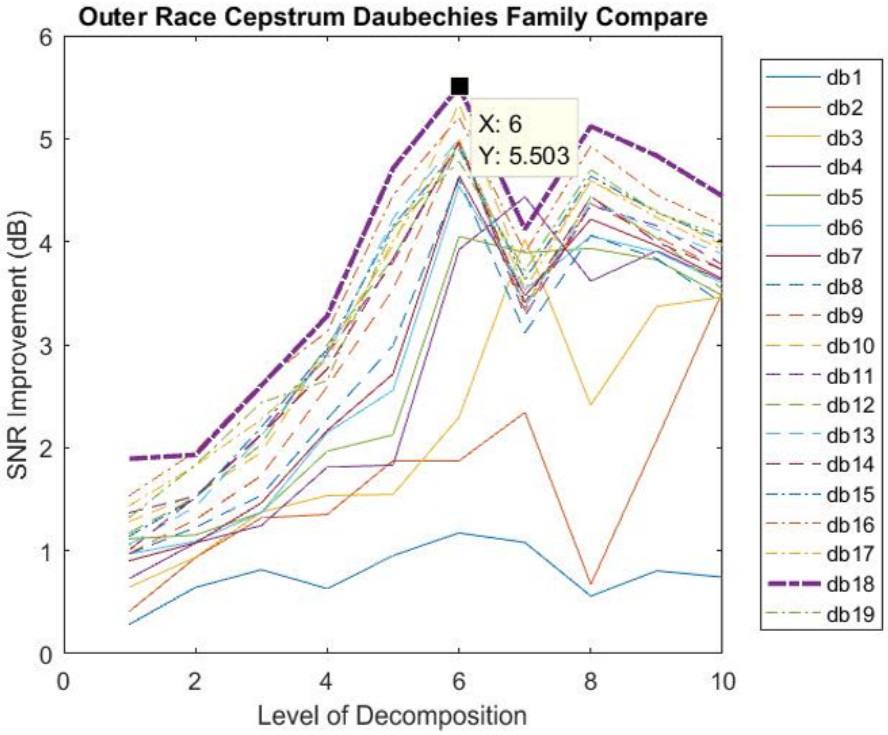

Figure 6.60: Short Time Signals - cepstrum analysis, outer race Daubechies family comparison.

6.4 Long Time Signals - Testbed Vibration Data

The long time signal results presented here have been summarised and tabulated rather than graphed. It was deemed impractical to present all 12 graphs for each of the 32 files analysed.

Tables 6.13, 6.14, 6.15 and 6.16 present the best performing wavelet de-noising parameters when the IMS 32 day testbed dataset was analysed using Fourier analysis.

Table 6.13: The best performing wavelet de-noising thresholds using Fourier analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	Level of wavelet decomposition	Threshold	SNR improvement
4 Mar	1	RigrSURE	0.00 dB
5 Mar	1	SqTwoLog	-0.01 dB
6 Mar	1	Minimaxi	-0.01 dB
7 Mar	1	RigrSURE	-0.06 dB
8 Mar	1	Minimaxi	-0.03 dB
9 Mar	2	SqTwoLog	-0.02 dB
10 Mar	1	RigrSURE	0.01 dB
11 Mar	1	RigrSURE	-0.05 dB
12 Mar	1	RigrSURE	-0.03 dB
13 Mar	1	RigrSURE	-0.01 dB
14 Mar	1	SqTwoLog	-0.07 dB
15 Mar	1	SqTwoLog	0.00 dB
16 Mar	1	RigrSURE	0.00 dB
17 Mar	1	RigrSURE	-0.04 dB
18 Mar	1	RigrSURE	0.00 dB
19 Mar	1	SqTwoLog	0.00 dB
20 Mar	1	Minimaxi	0.00 dB
21 Mar	1	Minimaxi	-0.03 dB
22 Mar	1	RigrSURE	-0.01 dB
23 Mar	1	RigrSURE	-0.06 dB
24 Mar	1	Minimaxi	-0.01 dB
25 Mar	1	Minimaxi	-0.04 dB
26 Mar	1	RigrSURE	0.00 dB
27 Mar	2	RigrSURE	-0.01 dB
28 Mar	1	RigrSURE	-0.02 dB
29 Mar	1	Minimaxi	-0.03 dB
30 Mar	1	SqTwoLog	0.00 dB
31 Mar	1	Minimaxi	-0.03 dB
1 Apr	1	HeurSURE	0.00 dB
2 Apr	1	HeurSURE	0.00 dB
3 Apr	1	Minimaxi	0.00 dB
4 Apr	1	SqTwoLog	-0.03 dB

Table 6.14: The best performing application of wavelet de-noising thresholds (soft or hard application) using Fourier analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Soft or Hard**	**SNR improvement**
4 Mar	1	Hard	0.00 dB
5 Mar	1	Soft	0.00 dB
6 Mar	1	Soft	0.00 dB
7 Mar	1	Soft	-0.03 dB
8 Mar	1	Soft	-0.01 dB
9 Mar	2	Soft	-0.01 dB
10 Mar	1	Hard	0.01 dB
11 Mar	1	Soft	-0.03 dB
12 Mar	1	Soft	-0.02 dB
13 Mar	1	Soft	-0.01 dB
14 Mar	1	Soft	-0.04 dB
15 Mar	1	Soft	0.00 dB
16 Mar	1	Soft	0.00 dB
17 Mar	1	Soft	-0.02 dB
18 Mar	1	Soft	0.00 dB
19 Mar	1	Hard	0.00 dB
20 Mar	1	Hard	0.00 dB
21 Mar	1	Soft	-0.02 dB
22 Mar	1	Soft	0.00 dB
23 Mar	1	Soft	-0.03 dB
24 Mar	1	Soft	-0.01 dB
25 Mar	1	Soft	-0.02 dB
26 Mar	1	Soft	0.00 dB
27 Mar	2	Soft	-0.01 dB
28 Mar	1	Soft	-0.01 dB
29 Mar	1	Soft	-0.02 dB
30 Mar	1	Soft	0.00 dB
31 Mar	1	Soft	-0.02 dB
1 Apr	1	Hard	0.01 dB
2 Apr	1	Hard	0.00 dB
3 Apr	1	Soft	0.00 dB
4 Apr	2	Soft	-0.01 dB

Table 6.15: The best performing wavelet de-noising noise estimation method using Fourier analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	Level of wavelet decomposition	Noise	SNR improvement
4 Mar	1	Mln and Sln	0.00 dB
5 Mar	1	One	-0.01 dB
6 Mar	1	One	-0.01 dB
7 Mar	1	Mln and Sln	-0.07 dB
8 Mar	1	Mln and Sln	-0.03 dB
9 Mar	1	One	-0.02 dB
10 Mar	1	Mln and Sln	0.01 dB
11 Mar	1	Mln and Sln	-0.05 dB
12 Mar	1	One	-0.03 dB
13 Mar	1	Mln and Sln	-0.01 dB
14 Mar	1	One	-0.07 dB
15 Mar	1	Mln and Sln	0.00 dB
16 Mar	1	One	0.00 dB
17 Mar	1	Mln and Sln	-0.05 dB
18 Mar	1	One	0.00 dB
19 Mar	1	One	0.00 dB
20 Mar	1	One	0.00 dB
21 Mar	1	One	-0.03 dB
22 Mar	1	One	-0.01 dB
23 Mar	1	One	-0.06 dB
24 Mar	1	One	-0.01 dB
25 Mar	1	Mln and Sln	-0.04 dB
26 Mar	1	Mln and Sln	-0.01 dB
27 Mar	2	Mln and Sln	-0.01 dB
28 Mar	1	One	-0.02 dB
29 Mar	1	One	-0.03 dB
30 Mar	1	One	0.00 dB
31 Mar	1	One	-0.03 dB
1 Apr	3	Sln	0.01 dB
2 Apr	1	One	0.00 dB
3 Apr	1	Mln and Sln	0.00 dB
4 Apr	2	One	-0.03 dB

Table 6.16: The best performing Daubechies wavelet for de-noising using Fourier analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Daubechies wavelet**	**SNR improvement**
4 Mar	3	DB3	0.01 dB
5 Mar	4	DB11	0.00 dB
6 Mar	4	DB7	0.02 dB
7 Mar	5	DB18	-0.06 dB
8 Mar	1	DB11	-0.03 dB
9 Mar	3	DB10	-0.02 dB
10 Mar	1	DB18	0.01 dB
11 Mar	4	DB10	-0.05 dB
12 Mar	4	DB5	0.00 dB
13 Mar	4	DB8	0.00 dB
14 Mar	3	DB4	-0.06 dB
15 Mar	4	DB7	0.04 dB
16 Mar	4	DB7	0.01 dB
17 Mar	4	DB6	-0.02 dB
18 Mar	4	DB5	0.01 dB
19 Mar	5	DB13	0.89 dB
20 Mar	4	DB5	0.03 dB
21 Mar	4	DB7	-0.02 dB
22 Mar	4	DB6	0.01 dB
23 Mar	3	DB3	-0.02 dB
24 Mar	4	DB3	0.04 dB
25 Mar	2	DB9	-0.04 dB
26 Mar	4	DB9	0.00 dB
27 Mar	3	DB4	-0.01 dB
28 Mar	4	DB6	-0.01 dB
29 Mar	2	DB8	-0.03 dB
30 Mar	4	DB15	0.00 dB
31 Mar	4	DB8	-0.02 dB
1 Apr	5	DB18	0.35 dB
2 Apr	5	DB17	0.15 dB
3 Apr	1	DB17	0.00 dB
4 Apr	5	DB16	0.02 dB

Tables 6.17, 6.18, 6.19 and 6.20 present the best performing wavelet de-noising parameters when the IMS 32 day testbed dataset was analysed using envelope analysis.

Table 6.17: The best performing wavelet de-noising thresholds using envelope analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	Level of wavelet decomposition	Threshold	SNR improvement
4 Mar	5	SqTwoLog	0.83 dB
5 Mar	1	RigrSURE	0.00 dB
6 Mar	1	SqTwoLog	0.00 dB
7 Mar	4	Minimaxi	0.03 dB
8 Mar	1	RigrSURE	0.00 dB
9 Mar	1	HeurSURE	0.00 dB
10 Mar	3	RigrSURE	0.00 dB
11 Mar	3	HeurSURE	0.00 dB
12 Mar	3	SqTwoLog	0.01 dB
13 Mar	1	RigrSURE	0.00 dB
14 Mar	1	SqTwoLog	0.00 dB
15 Mar	3	SqTwoLog	0.01 dB
16 Mar	3	SqTwoLog	0.03 dB
17 Mar	1	RigrSURE	0.00 dB
18 Mar	1	RigrSURE	0.00 dB
19 Mar	4	SqTwoLog	0.02 dB
20 Mar	3	SqTwoLog	0.01 dB
21 Mar	2	Minimaxi	0.00 dB
22 Mar	2	SqTwoLog	0.00 dB
23 Mar	3	RigrSURE	0.01 dB
24 Mar	4	SqTwoLog	0.12 dB
25 Mar	4	SqTwoLog	0.11 dB
26 Mar	1	RigrSURE	0.00 dB
27 Mar	3	RigrSURE	0.01 dB
28 Mar	2	SqTwoLog	0.01 dB
29 Mar	2	Minimaxi	0.00 dB
30 Mar	4	Minimaxi	0.00 dB
31 Mar	3	SqTwoLog	0.03 dB
1 Apr	3	SqTwoLog	0.01 dB
2 Apr	3	RigrSURE	0.00 dB
3 Apr	4	SqTwoLog	0.07 dB
4 Apr	1	Minimaxi	0.00 dB

Table 6.18: The best performing application of wavelet de-noising thresholds (soft or hard application) using envelope analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Soft or Hard**	**SNR improvement**
4 Mar	5	Hard	0.69 dB
5 Mar	1	Hard	0.00 dB
6 Mar	1	Hard	0.00 dB
7 Mar	4	Hard	0.02 dB
8 Mar	1	Soft	0.00 dB
9 Mar	1	Soft	0.00 dB
10 Mar	3	Hard	0.00 dB
11 Mar	3	Soft	0.00 dB
12 Mar	3	Hard	0.01 dB
13 Mar	1	Soft	0.00 dB
14 Mar	3	Hard	0.00 dB
15 Mar	3	Hard	0.01 dB
16 Mar	3	Hard	0.02 dB
17 Mar	1	Soft	0.00 dB
18 Mar	1	Soft	0.00 dB
19 Mar	4	Hard	0.02 dB
20 Mar	3	Hard	0.01 dB
21 Mar	2	Hard	0.00 dB
22 Mar	2	Hard	0.00 dB
23 Mar	3	Hard	0.01 dB
24 Mar	4	Hard	0.09 dB
25 Mar	4	Hard	0.07 dB
26 Mar	1	Soft	0.00 dB
27 Mar	3	Hard	0.01 dB
28 Mar	2	Hard	0.01 dB
29 Mar	2	Hard	0.00 dB
30 Mar	1	Soft	0.00 dB
31 Mar	3	Hard	0.02 dB
1 Apr	3	Hard	0.01 dB
2 Apr	3	Hard	0.00 dB
3 Apr	4	Hard	0.06 dB
4 Apr	1	Soft	0.00 dB

Table 6.19: The best performing wavelet de-noising noise estimation method using envelope analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Noise**	**SNR improvement**
4 Mar	5	One	0.87 dB
5 Mar	1	Mln and Sln	0.00 dB
6 Mar	1	One	0.00 dB
7 Mar	4	Mln	0.02 dB
8 Mar	1	Mln and Sln	0.00 dB
9 Mar	1	One	0.00 dB
10 Mar	3	Mln	0.00 dB
11 Mar	3	One	0.00 dB
12 Mar	3	Mln	0.01 dB
13 Mar	1	Mln and Sln	0.00 dB
14 Mar	3	Mln	0.00 dB
15 Mar	3	Mln	0.01 dB
16 Mar	3	One	0.03 dB
17 Mar	1	Mln and Sln	0.00 dB
18 Mar	1	Mln and Sln	0.00 dB
19 Mar	4	One	0.02 dB
20 Mar	3	Mln	0.01 dB
21 Mar	2	One	0.00 dB
22 Mar	2	One	0.01 dB
23 Mar	3	Mln	0.01 dB
24 Mar	4	One	0.13 dB
25 Mar	4	One	0.11 dB
26 Mar	5	One	0.05 dB
27 Mar	3	Mln	0.01 dB
28 Mar	2	Mln	0.01 dB
29 Mar	2	Mln	0.00 dB
30 Mar	1	Mln and Sln	0.00 dB
31 Mar	3	One	0.03 dB
1 Apr	3	One	0.02 dB
2 Apr	3	Sln	0.00 dB
3 Apr	4	One	0.06 dB
4 Apr	1	Mln and Sln	0.00 dB

Table 6.20: The best performing Daubechies wavelet for de-noising using envelope analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Daubechies wavelet**	**SNR improvement**
4 Mar	5	DB1	1.79 dB
5 Mar	2	DB2	0.01 dB
6 Mar	4	DB1	0.13 dB
7 Mar	4	DB1	0.30 dB
8 Mar	5	DB1	1.02 dB
9 Mar	5	DB1	0.09 dB
10 Mar	5	DB16	0.33 dB
11 Mar	5	DB8	0.26 dB
12 Mar	5	DB1	0.74 dB
13 Mar	5	DB4	0.13 dB
14 Mar	4	DB3	0.27 dB
15 Mar	5	DB2	0.49 dB
16 Mar	5	DB1	0.65 dB
17 Mar	5	DB18	0.17 dB
18 Mar	5	DB17	0.23 dB
19 Mar	4	DB2	0.41 dB
20 Mar	4	DB3	0.05 dB
21 Mar	4	DB5	0.05 dB
22 Mar	4	DB2	0.06 dB
23 Mar	5	DB16	0.67 dB
24 Mar	4	DB1	1.35 dB
25 Mar	5	DB2	1.79 dB
26 Mar	5	DB8	0.54 dB
27 Mar	3	DB1	0.11 dB
28 Mar	4	DB2	0.48 dB
29 Mar	4	DB4	0.10 dB
30 Mar	4	DB2	0.20 dB
31 Mar	4	DB1	0.66 dB
1 Apr	5	DB1	0.74 dB
2 Apr	5	DB13	0.46 dB
3 Apr	5	DB1	0.37dB
4 Apr	5	DB1	0.61 dB

Tables 6.21, 6.22, 6.23 and 6.24 present the best performing wavelet de-noising parameters when the IMS 32 day testbed dataset was analysed using envelope analysis.

Table 6.21: The best performing wavelet de-noising thresholds using cepstrum analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	Level of wavelet decomposition	Threshold	SNR improvement
4 Mar	6	SqTwoLog	21.32 dB
5 Mar	6	SqTwoLog	23.17 dB
6 Mar	6	SqTwoLog	26.50 dB
7 Mar	6	SqTwoLog	24.97 dB
8 Mar	6	SqTwoLog	24.91 dB
9 Mar	6	SqTwoLog	27.32 dB
10 Mar	6	SqTwoLog	26.18 dB
11 Mar	6	SqTwoLog	27.46 dB
12 Mar	6	SqTwoLog	27.20 dB
13 Mar	6	SqTwoLog	25.18 dB
14 Mar	6	SqTwoLog	27.81 dB
15 Mar	6	SqTwoLog	26.85 dB
16 Mar	6	SqTwoLog	27.46 dB
17 Mar	6	SqTwoLog	26.42 dB
18 Mar	6	SqTwoLog	28.14 dB
19 Mar	6	SqTwoLog	28.27 dB
20 Mar	6	SqTwoLog	27.20 dB
21 Mar	6	SqTwoLog	23.36 dB
22 Mar	6	SqTwoLog	27.88 dB
23 Mar	6	SqTwoLog	24.27 dB
24 Mar	6	SqTwoLog	27.11 dB
25 Mar	7	SqTwoLog	21.70 dB
26 Mar	6	SqTwoLog	30.92 dB
27 Mar	6	SqTwoLog	26.47 dB
28 Mar	6	SqTwoLog	27.55 dB
29 Mar	6	SqTwoLog	23.63 dB
30 Mar	6	SqTwoLog	26.78 dB
31 Mar	6	SqTwoLog	26.23 dB
1 Apr	6	SqTwoLog	28.54 dB
2 Apr	6	SqTwoLog	27.15 dB
3 Apr	6	SqTwoLog	24.37 dB
4 Apr	6	SqTwoLog	26.86 dB

Table 6.22: The best performing application of wavelet de-noising thresholds (soft or hard application) using cepstrum analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Soft or Hard**	**SNR improvement**
4 Mar	6	Hard	16.32 dB
5 Mar	6	Hard	16.27 dB
6 Mar	6	Hard	18.93 dB
7 Mar	6	Hard	19.73 dB
8 Mar	6	Hard	16.80 dB
9 Mar	6	Hard	18.36 dB
10 Mar	6	Hard	18.81 dB
11 Mar	6	Hard	20.18 dB
12 Mar	6	Hard	20.19 dB
13 Mar	6	Hard	17.91 dB
14 Mar	6	Hard	18.53 dB
15 Mar	6	Hard	18.30 dB
16 Mar	6	Hard	18.75 dB
17 Mar	6	Hard	17.86 dB
18 Mar	6	Hard	20.51 dB
19 Mar	6	Hard	18.01 dB
20 Mar	6	Hard	19.85 dB
21 Mar	6	Hard	16.48 dB
22 Mar	6	Hard	19.78 dB
23 Mar	6	Hard	18.00 dB
24 Mar	6	Hard	19.84 dB
25 Mar	6	Hard	15.90 dB
26 Mar	6	Hard	22.48 dB
27 Mar	6	Hard	18.69 dB
28 Mar	6	Hard	18.77 dB
29 Mar	6	Hard	17.07 dB
30 Mar	6	Hard	20.68 dB
31 Mar	6	Hard	19.61 dB
1 Apr	6	Hard	19.10 dB
2 Apr	6	Hard	18.00 dB
3 Apr	6	Hard	18.03 dB
4 Apr	6	Hard	18.97 dB

Table 6.23: The best performing wavelet de-noising noise estimation method using cepstrum analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	Level of wavelet decomposition	Noise	SNR improvement
4 Mar	6	One	36.77 dB
5 Mar	6	One	33.60 dB
6 Mar	6	One	38.82 dB
7 Mar	6	One	37.22 dB
8 Mar	6	One	32.64 dB
9 Mar	6	One	34.86 dB
10 Mar	6	One	37.63 dB
11 Mar	6	One	37.46 dB
12 Mar	6	One	37.05 dB
13 Mar	6	One	38.66 dB
14 Mar	6	One	34.63 dB
15 Mar	6	One	35.07 dB
16 Mar	6	One	36.41 dB
17 Mar	6	One	34.39 dB
18 Mar	6	One	38.05 dB
19 Mar	6	One	33.78 dB
20 Mar	6	One	35.08 dB
21 Mar	6	One	32.91 dB
22 Mar	6	One	37.52 dB
23 Mar	6	One	36.26 dB
24 Mar	6	One	35.78 dB
25 Mar	6	One	30.48 dB
26 Mar	6	One	39.38 dB
27 Mar	6	One	37.52 dB
28 Mar	6	One	36.00 dB
29 Mar	6	One	34.43 dB
30 Mar	6	One	38.17 dB
31 Mar	6	One	35.99 dB
1 Apr	6	One	36.64 dB
2 Apr	6	One	37.04 dB
3 Apr	6	One	36.07 dB
4 Apr	6	One	37.17 dB

Table 6.24: The best performing Daubechies wavelet for de-noising using cepstrum analysis, applied to the IMS 32 day testbed dataset.

Date the data file was acquired	**Level of wavelet decomposition**	**Daubechies wavelet**	**SNR improvement**
4 Mar	6	DB2	20.95 dB
5 Mar	6	DB17	22.07 dB
6 Mar	6	DB17	24.50 dB
7 Mar	6	DB12	29.56 dB
8 Mar	6	DB12	22.75 dB
9 Mar	8	DB13	24.38 dB
10 Mar	6	DB14	23.06 dB
11 Mar	6	DB19	25.33 dB
12 Mar	6	DB12	24.59 dB
13 Mar	6	DB17	21.21 dB
14 Mar	9	DB6	22.44 dB
15 Mar	8	DB15	23.58 dB
16 Mar	8	DB10	25.23 dB
17 Mar	8	DB17	21.10 dB
18 Mar	6	DB8	25.18 dB
19 Mar	8	DB19	24.38 dB
20 Mar	6	DB14	25.38 dB
21 Mar	6	DB18	22.81 dB
22 Mar	6	DB19	25.41 dB
23 Mar	6	DB14	28.20 dB
24 Mar	6	DB17	25.92 dB
25 Mar	8	DB10	23.25 dB
26 Mar	6	DB17	27.17 dB
27 Mar	6	DB3	24.24 dB
28 Mar	6	DB19	25.89 dB
29 Mar	6	DB17	22.72 dB
30 Mar	6	DB17	24.38 dB
31 Mar	6	DB14	24.51 dB
1 Apr	6	DB3	24.38 dB
2 Apr	8	DB19	24.45 dB
3 Apr	6	DB13	23.12 dB
4 Apr	6	DB13	23.38 dB

Chapter 7

Analysis

7.1 Chapter Overview

The main aim of this dissertation is to provide earlier detection of a rolling element bearing fault by improving the signal to noise ratio (SNR) of a bearing vibration fault signal. The method chosen to improve the SNR is based on wavelet de-noising and there are many options available for configuring the wavelet de-noising process. This work focuses on the range of de-noising parameters available within the MATLAB^{TM} computing environment.

Three groups of signals have been tested to determine which wavelet de-noising parameters are most suited to improving the SNR for bearing vibration signals. The signal groups are:

- Simulation signals.
- Short time signals acquired on a bearing testbed arrangement.
- Long time signal acquired on a testbed arrangement but recorded over a period of 32 days.

The wavelet de-noising process consists of five variables.

- The type of threshold to be used.
- The manner in which the threshold is applied.

- The noise scale estimation.
- The range of wavelets available within a given wavelet family.
- The level of wavelet decomposition.

When conducting the analysis of best performing wavelet de-noising parameters, three characteristic fault frequency (CFF) extraction methods were used. The three extraction methods were Fourier analysis, envelope analysis and cepstrum analysis and the results for each method were grouped into four categories. The first category compared the four threshold types. The second category compared either a soft or hard application of the threshold. The third category compared the MATLAB^{TM} noise scale estimation options. The fourth category compared the choice of wavelets available within the Daubechies wavelet family.

Each of the four categories used the level of wavelet decomposition as a independent variable of sorts. Changes in SNR for a de-noising variable were tracked against the level of decomposition. This is shown in the graphs in Chapter 6, Section 6.3 where the level of decomposition is used as the X axis on the graphs.

However when presenting the results for the analysis of the simulation signals and the long time signals, the parameters that most improved the SNR have been taken from the graphs and tabulated. The simulation signals group and the long time signals group are a series of sequential data files. The simulation signals consist of nine files that vary the magnitude of Gaussian added noise (0 to 0.8 magnitude). The best performing wavelet de-noising parameters for the simulation signals are tabulated against the level of added Gaussian noise. The long time signals consist of 32 files that track bearing wear over a period of 32 days. The best performing wavelet de-noising parameters for the long time signals are tabulated against the day they were acquired.

Before the analysis of the results begins it is important to note the differences in the relative frequency bandwidths being observed between the simulation signals, and the short time and long time signals. The simulation signal has frequency components at 1611 Hz therefore the bandwidth of frequencies being observed has been extended to 2000 Hz. The SNR is being calculated over this full range where noise is considered all the 'other' frequencies not part of the desired signal. Contrast this to the short time and long time signals where the maximum frequency being observed is 250 Hz and 300 Hz

respectively. In terms of the SNR calculation the smaller bandwidth being observed will contribute less to the noise component of the SNR. It is for this reason that a straight comparison of SNR between the three signal groups cannot be undertaken. In any case, this work is about finding the best performing wavelet de-noising parameters within each signal group and then making comparison between those.

7.2 Simulation Signals

7.2.1 Fourier Analysis

Fourier analysis of the noise free simulation signal produces a frequency spectrum with a dominant frequency component at 1611 Hz. Wavelet de-noising of the noisy simulation signal attempts to improve the SNR of the 1611 Hz component with respect to the remaining frequencies in the signal. The results of the Fourier analysis of the simulation signals are shown in Tables 6.1, 6.2, 6.3, and 6.4.

Table 6.1 demonstrates that varying the wavelet de-noising threshold type produces no gain in SNR for the simulation signal and in most cases the SNR is reduced by a small margin. The table shows level one was the most effective level of decomposition when de-noising the signal, all other levels of decomposition produced greater reductions in the SNR. In most cases all four thresholds produced the same reduction in SNR and where a particular threshold was selected as the most effective, it was only by a small margin.

Table 6.2 demonstrates the soft or hard application of the wavelet de-noising threshold produced no gain in the SNR of the simulation signal. In all cases the soft application of the threshold was the most effective, and this occurred at the first level of decomposition for all magnitudes of added Gaussian noise.

Table 6.3 demonstrates the Mln and Sln methods of noise scale estimation were equally the most effective options for de-noising the simulation signal however, for all magnitudes of Gaussian noise a SNR loss was realised during the analysis.

Table 6.4 shows minor SNR improvements were realised when de-noising using Daubechies wavelets DB9, DB10 and DB11. DB11 was most commonly the best wavelet. The SNR gains were all achieved at the second level of decomposition.

Figures 7.1 and 7.2 show very little change in the frequency spectrum when the optimum wavelet de-noising parameters of RigrSURE threshold, soft threshold application, Sln noise estimation, level 2 decomposition and Daubechies wavelet DB11 selected. Looking at Figure 7.1 it is clear that the frequency content making up the simulation signal exists across a wide band of frequencies. The wavelet de-noising process has little effect on removing these frequencies since they are part of the noise free signal. In this case, since few of these frequencies are removed then the noise component of the SNR is largely left unchanged from de-noising and hence only very small changes in the SNR result.

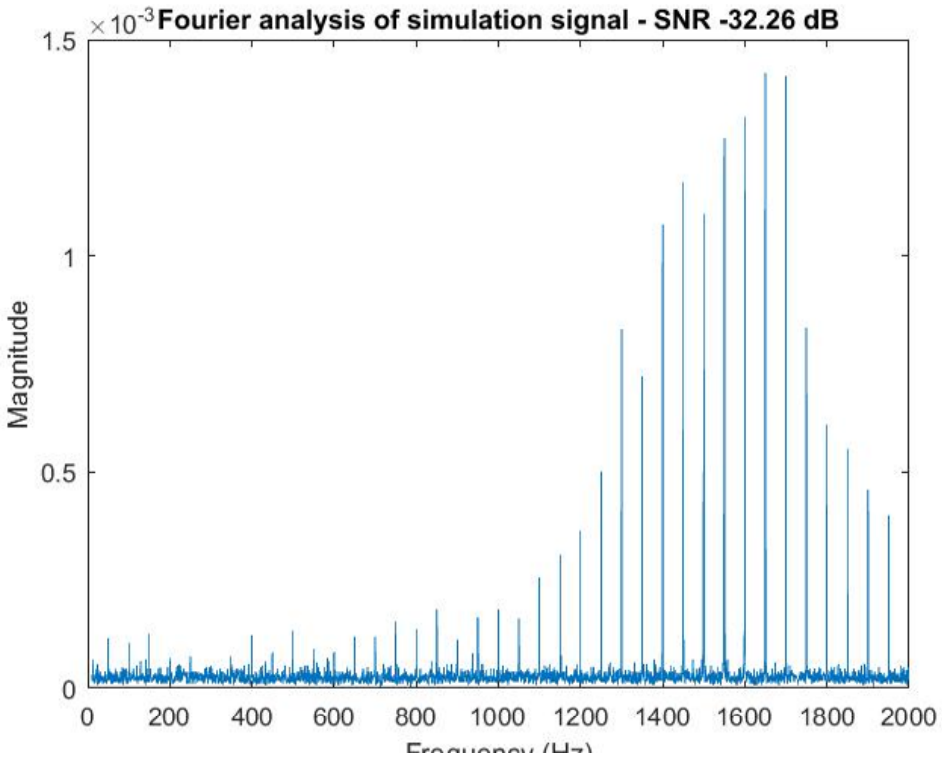

Figure 7.1: Fourier analysis of the simulation signal with 0.5 magnitude Gaussian added noise.

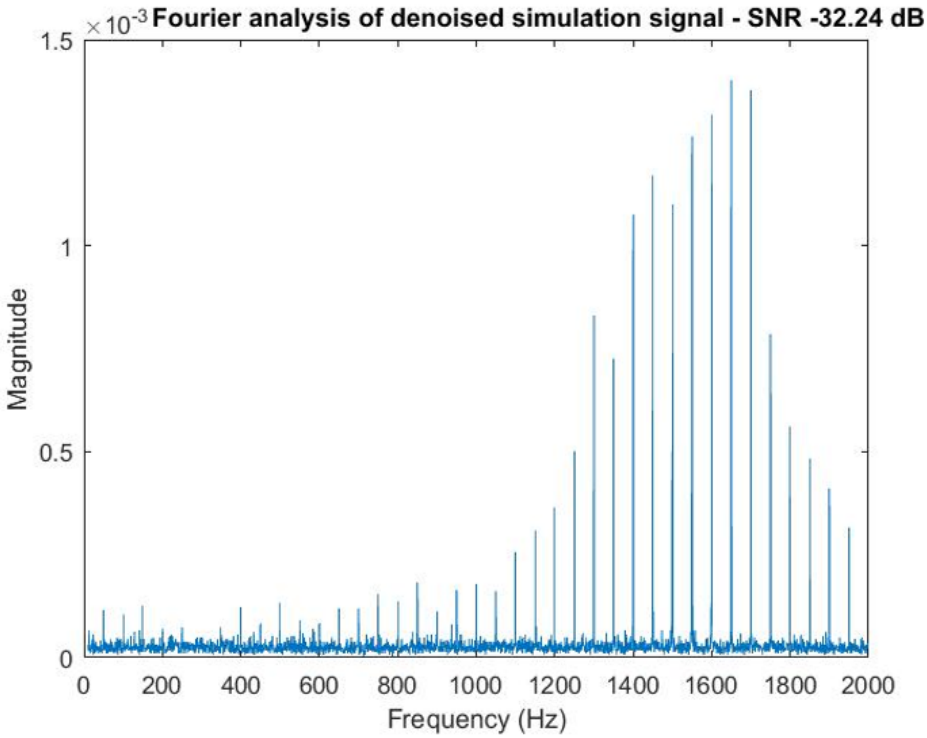

Figure 7.2: Fourier analysis of the de-noised simulation signal with 0.5 magnitude Gaussian noise added. The wavelet de-noising parameters used here were RigrSURE, Soft, Sln, level 2 and DB11.

7.2.2 Envelope Analysis

Envelope analysis of the noise free simulation signal produces a frequency spectrum with a dominant frequency component at 50 Hz. Wavelet de-noising of the noisy simulation signal attempts to improve the SNR of the 50 Hz component with respect to the remaining frequencies in the signal. The results of the envelope analysis of the simulation signals are shown in Tables 6.5, 6.6, 6.7, and 6.8.

Table 6.5 shows the most effective threshold used was the universal threshold (SqTwoLog). The table shows the 7th level of decomposition was the best option and combined with the universal threshold produced an average SNR improvement of 29.71 dB.

Table 6.6 shows that hard application of the threshold is the best option for de-noising the simulation signal. The 7th level of wavelet decomposition was the best performing parameter combining with the hard application of threshold to produce an average SNR improvement of 22.49 dB over the range of simulation signals.

Table 6.7 shows the noise scale estimation method of 'One' provides the best increase in SNR when using envelope analysis on the simulation signal.

Table 6.8 shows Daubechies wavelet DB18 commonly out performs other wavelets in improving the SNR.

Figures 7.3 and 7.4 show a large increase in the SNR as a result of wavelet de-noising. The de-noising parameters were universal threshold (SqTwoLog), hard application of threshold, One noise estimation, level 7 decomposition and Daubechies wavelet DB18. Figure 7.4 shows most of the frequency content has been removed and a large 50 Hz frequency component remains. This is a result of the lower frequency 50 Hz pulse being demodulated from the higher frequency 1611 - 1666 Hz frequency components (and surrounding beat frequencies) that make up the oscillation in the simulation signal. The removal of the large range of frequencies has the effect of reducing the noise component of the SNR by a substantial amount, resulting in the large SNR gains shown in the tables.

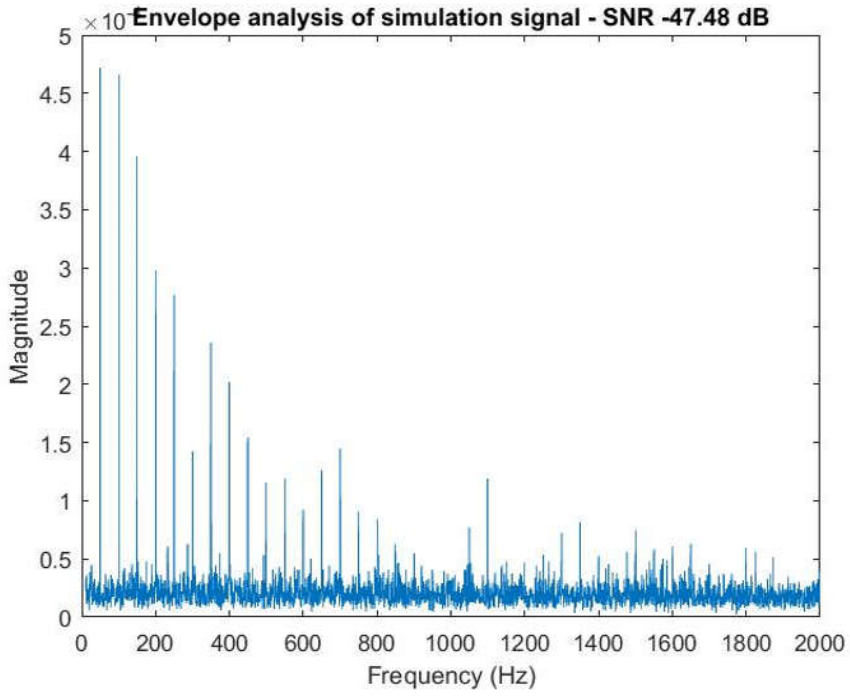

Figure 7.3: Envelope analysis of the simulation signal with 0.5 magnitude Gaussian added noise.

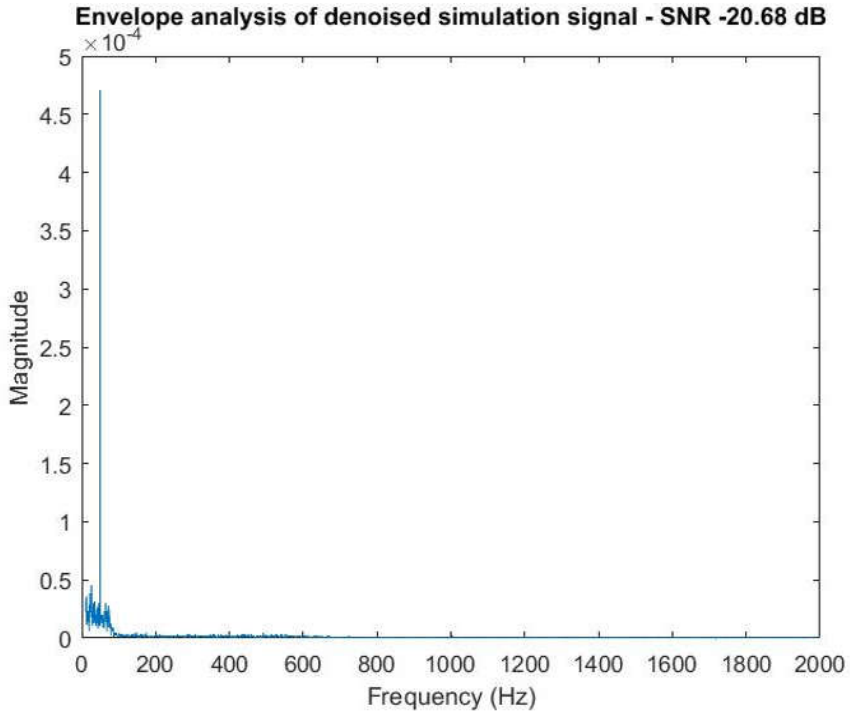

Figure 7.4: Envelope analysis of the de-noised simulation signal with 0.5 magnitude Gaussian noise added. The wavelet de-noising parameters used here were SqTwoLog, Hard, One, level 7 and DB18.

7.2.3 Cepstrum Analysis

Cepstrum analysis of the noise free simulation signal produces a quefrency spectrum with a dominant component at 20.05 ms. Wavelet de-noising of the noisy simulation signal attempts to improve the SNR of the 20.05 ms component with respect to the remaining quefrencies in the signal. The results of the cepstrum analysis of the simulation signals are shown in Tables 6.9, 6.10, 6.11, and 6.12.

Table 6.9 shows the most effective threshold was the universal threshold (SqTwoLog) at wavelet decomposition level 2. The universal threshold was the best performing threshold five times out of the nine different noise magnitudes assessed.

Table 6.10 shows mixed results for the best application of threshold. For signals with stronger initial SNR the soft application of threshold is the best choice but for the lower SNR before de-noising, the hard application of threshold was the best option, combined with level 2 decomposition. Hard thresholding realised an average SNR gain of 4.21 dB.

Table 6.11 shows the noise scale estimation of One most commonly performed the best and produced an average SNR gain of 5.82 dB. The second level of decomposition was most commonly selected as the best option.

Table 6.12 shows mixed results when selecting the best performing Daubechies wavelet. Daubechies wavelet DB4 and DB5 perform well in the mid-to-low initial SNR ranges. DB4 averages a SNR gain of 10.00 dB. The third level of decomposition featured the most times as the most successful.

Figures 7.5 and 7.6 show moderate increases in the SNR as a result of wavelet de-noising. The de-noising parameters were universal threshold (SqTwoLog), hard application of threshold, One noise estimation, level 2 decomposition and Daubechies wavelet DB4. Note that figures use different scale magnitudes on the Y axis. Although the background noise level on Figure 7.6 appears lower than Figure 7.5 it is actually the change in signal component magnitude that is increasing the SNR.

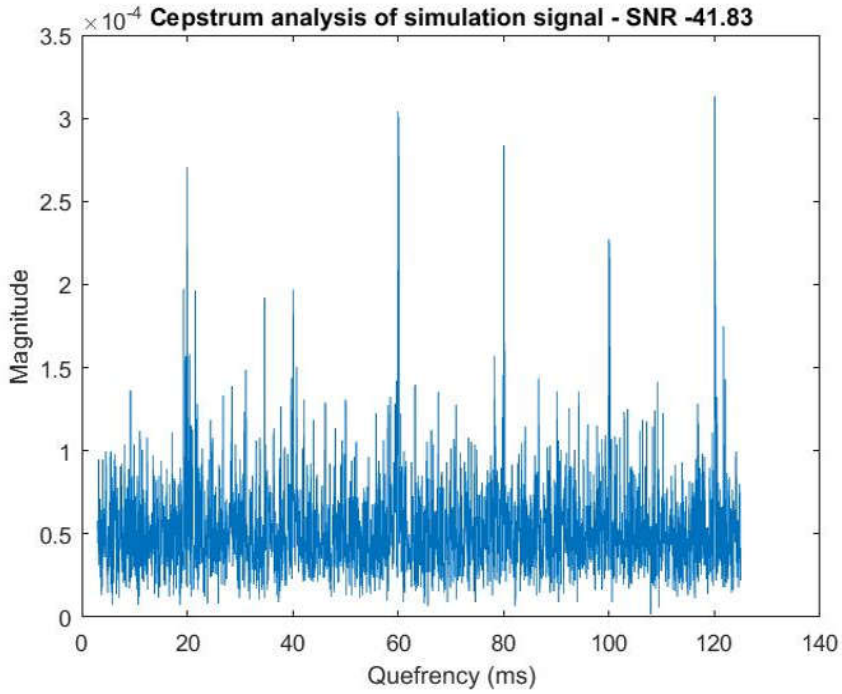

Figure 7.5: Cepstrum analysis of the simulation signal with 0.5 magnitude Gaussian added noise.

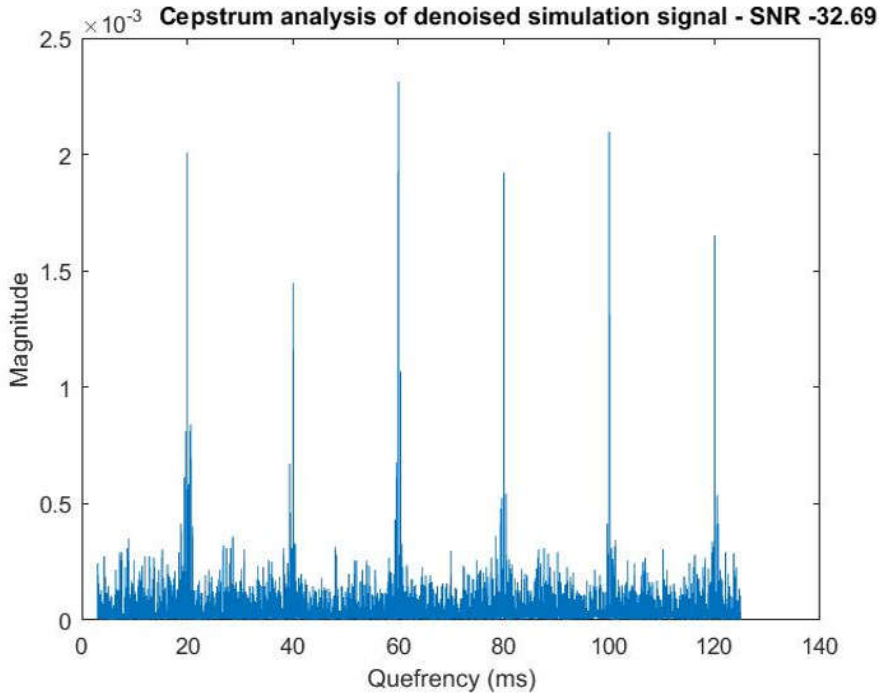

Figure 7.6: Cepstrum analysis of the denoised simulation signal with 0.5 magnitude Gaussian noise added. The wavelet de-noising parameters used here were SqTwoLog, Hard, One, level 2 and DB4.

7.2.4 Conclusion

Fourier analysis of the wavelet de-noised simulation signal resulted in very small gains in SNR compared to the original noisy signal. The frequency content of the simulation signal consists of a wide band of frequencies and the wavelet de-noising process did not alter these much. This resulted in small changes in the overall SNR.

Wavelet de-noising of the simulation signals allowed the envelope analysis process to realise large gains in SNR when compared to the noisy signal. Wavelet de-noising had the effect of attenuating the frequency components contributing to noise and allowed the envelope analysis to extract the lower frequency 50 Hz component from the signal.

Ceptrsum analysis combined with wavelet de-noising resulted in SNR gains in the 6-10 dB range. The wavelet de-noising process did not appear to reduce the noise levels in the quefrency domain but rather improved the SNR by increasing the strength of the signal components.

7.3 Short Time Signals

The results for the short time signals are the product of 70 bearing vibration files analysed. The 70 different files come from three different sources and can be grouped into four areas, rolling element faults, inner race faults, outer race faults and fault free normal bearings. Since each file is analysed using Fourier analysis, envelope analysis and cepstrum analysis a very large group of result graphs were produced.

The large range of graphs have been condensed down into the four groups just described and are presented in Chapter 6. One reason for doing this is to make presenting the information easier. The main reason for combining the files into the groups is to average out the results and identify the wavelet de-noising parameters that *on average* performed the best. In this study some bearing signals responded to wavelet de-noising differently than others so analysing a single file and drawing a conclusion about the best de-noising parameters would be inappropriate. By averaging out and summarising the results it is possible that any future reader of this work may use the wavelet de-noising parameters recommended at the end of this chapter, de-noise a vibration signal from a different type of rolling element bearing, and expect the de-noising process to be successful.

An explanation of the summarised results from the short time signals analysis follows.

7.3.1 Fourier Analysis

Issue with higher levels of wavelet decomposition

The combined results of the Fourier analysis of short time signals acquired from various testbed arrangements are shown in Figures 6.1, 6.2, 6.3 and 6.4. Figure 6.1 compares the four thresholds, Figure 6.2 compares either soft or hard application of the threshold, Figure 6.3 compares the three noise scale estimation methods and Figure 6.4 compares the 19 Daubechies wavelets. Note that all four figures display a similar shape as the level of decomposition changes from 1 to 10. The change in SNR remains relatively flat from decomposition levels one to five. Above decomposition level five the SNR curves in all four figures dip sharply down and then rise up again as the level of decomposition increases.

When observing some of the Fourier analysis figures relating to specific bearing faults such as inner race faults, the SNR curves follow similar shapes. In some cases, such as Figure 6.49 which shows the threshold comparison for bearing outer race faults, the higher levels of decomposition produce gains in SNR.

The characteristic dip in the SNR was investigated to assess why decomposition levels six and seven consistently reduced the SNR. It was identified that at these levels of wavelet decomposition the main frequency components representing the desired bearing fault signal were attenuated in such a way that the original signal was removed. Higher levels of decomposition appear to attenuate the frequency components such that the remaining components are concentrated around the low frequency band of 0-50 Hz. To illustrate this concept refer to Figures 7.7 and 7.8. Figure 7.7 shows the frequency spectrum of a wavelet de-noised signal that has been decomposed to level 5. The signal is from a bearing with an outer race fault and has a CFF at 150 Hz. Figure 7.8 shows the frequency spectrum of the same signal but now decomposed to level 6. Note the 150 Hz component is completely removed however the measure of SNR from this spectrum showed an increase in SNR of 13.42 dB.

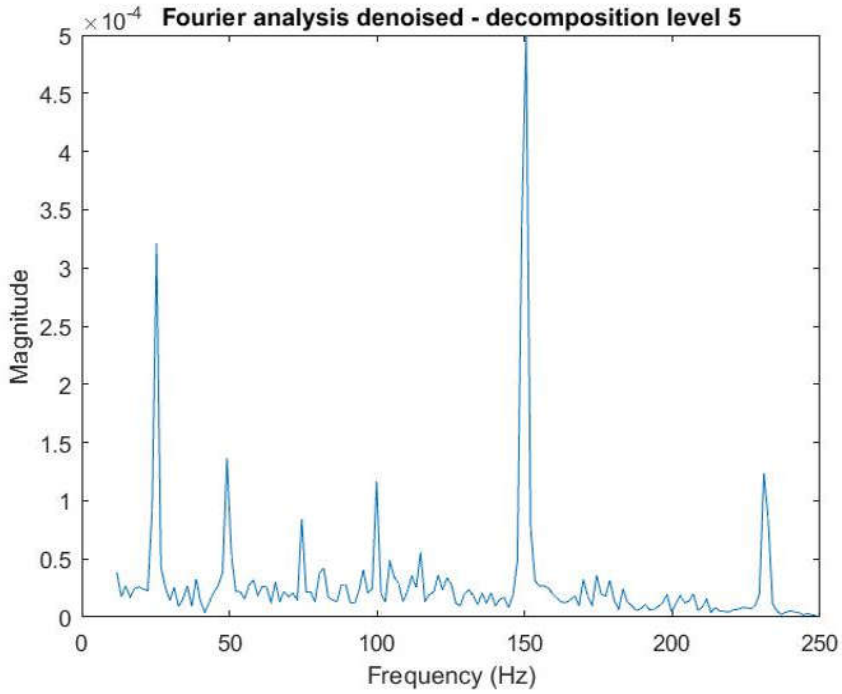

Figure 7.7: Frequency spectrum of a wavelet de-noised signal decomposed to level 5.

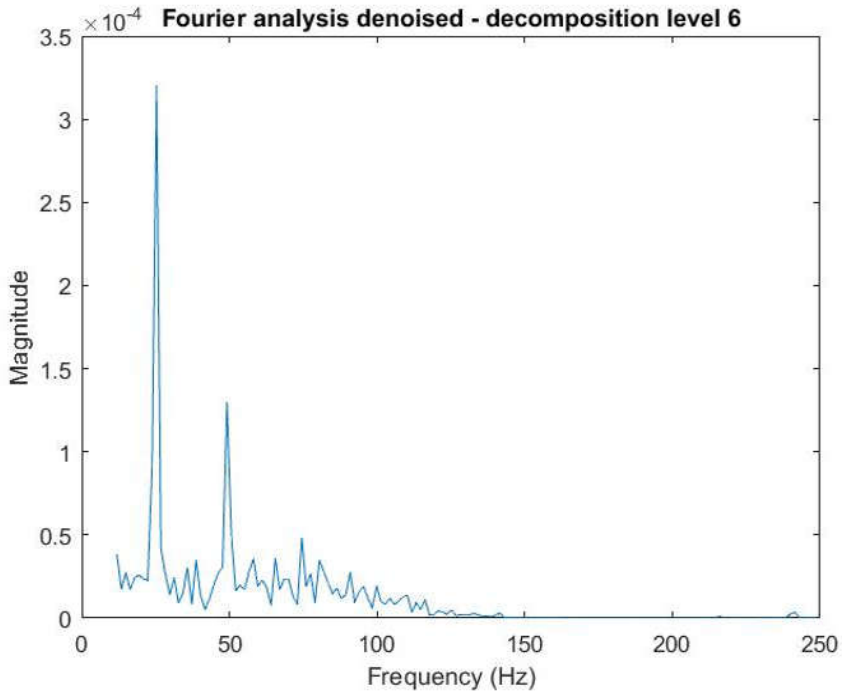

Figure 7.8: Frequency spectrum of a wavelet de-noised signal decomposed to level 6.

These figures explain why the SNR drops sharply at wavelet decomposition levels around six and seven but they do not explain why the SNR improves again. The higher levels of decomposition were analysed and Figure 7.9 shows almost all frequency components above 50 Hz have been attenuated to near zero. The small band of remaining frequencies are concentrated between 0 and 30 Hz. Within this band is the cage frequency at 14 Hz. The reason for the SNR increase is that signal energy in concentrated on or around the 14 Hz band and everywhere else in the spectrum the frequency components are almost zero. In terms of the SNR equation, the signal energy component has relatively strong energy at 14 Hz and the noise energy component is much less.

A number of short time signals have been analysed to confirm that the signal begins increased attenuation at wavelet decomposition levels above level 5. These results are not included in this work however is has been confirmed that this is the case for all short time signals. For this reason analysis of the Fourier results will only assess wavelet decomposition levels up to an including level 5.

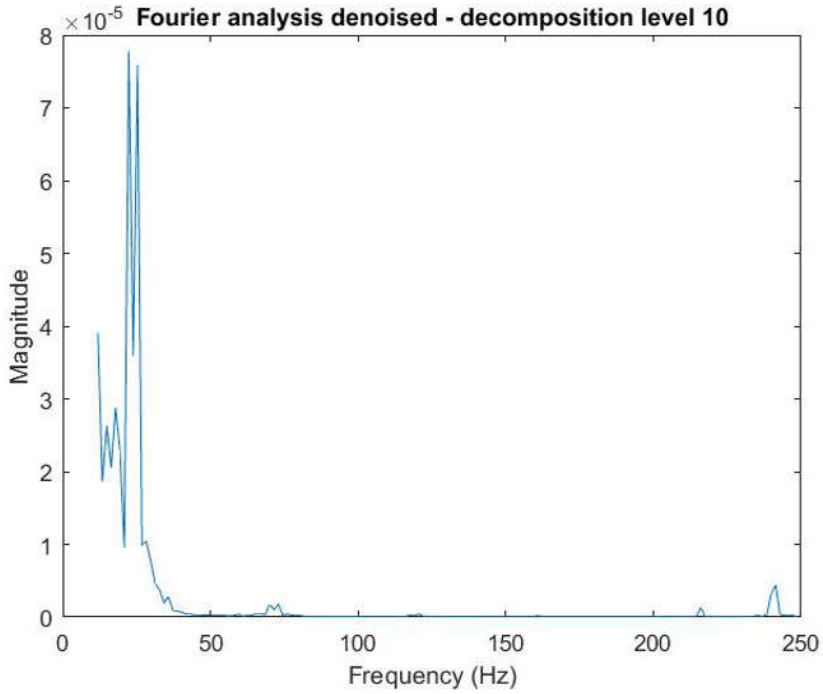

Figure 7.9: Frequency spectrum of a wavelet de-noised signal decomposed to level 10.

Fourier analysis up to decomposition level 5 only

Figure 6.1 shows on average the best performing threshold was the universal threshold (SqTwoLog) and was most effective at the 1st level of wavelet decomposition, although the average SNR change was a gain of 0.002 dB. When looking at specific bearing faults the universal threshold (SqTwoLog) was the most effective and always showed a peak SNR change at the 1st level of decomposition.

Figure 6.2 shows the soft application of threshold was the most effective and had the best response at the 1st level of decomposition and the change in SNR was 0.001 dB. Analysis of Figures 6.25 and 6.37 also supports this claim showing the soft application of threshold as the best option, also at the 1st level of decomposition.

Figure 6.3 shows the best performing noise scale estimation was One and occurred at decomposition level one. The SNR improvement at this level was 0.001 dB however the outer race bearing noise scale estimation suggests the optimum level of decomposition was level 5 and produced a SNR improvement of 0.61 dB, as shown in Figure 6.51.

Figure 6.4 shows the comparison between all the Daubechies wavelets for Fourier analysis. Daubechies wavelet DB19 was the best performing and had a peak improvement of 1.72 dB at the 5th level of decomposition. This is strongly supported in Figures 6.16, 6.28, 6.40 and 6.52 where the specific bearing fault graphs show DB19 at the 5th level of decomposition to be the most effective Daubechies wavelet.

7.3.2 Envelope Analysis

Issue with higher levels of wavelet decomposition

A similar issue to that identified in Section 7.3.1 has also been discovered with envelope analysis in relation to the level of wavelet decomposition and its effect on the SNR when de-noising. In this instance the situation is slightly different however, as it appears that only certain combinations of wavelet parameters are attenuating the signal and grouping signal energy around the cage frequency. Further, the issue only applies to the Case Western Reserve bearing dataset signals acquired from bearings with rolling element faults and some outer race faults. It is understood that the false increase in SNR from these bearings

will only slightly skew the average SNR improvement and can therefore be overlooked in this analysis.

Envelope analysis

The combined results of the envelope analysis of short time signals acquired from various testbed arrangements are shown in Figures 6.5, 6.6, 6.7 and 6.8.

Figure 6.5 shows on average the best performing de-noising threshold was the universal threshold (SqTwoLog) and was most effective at the 9th level of decomposition. The averaged value for SNR improvement was 11.35 dB. This is supported by the results for the envelope threshold comparisons for normal bearings, rolling element fault, inner race faults and outer race faults (Figures 6.17, 6.17, 6.41 and 6.53). As noted above, the 9th level of decomposition produces larger than usual increases in SNR for the rolling element comparison with a peak SNR improvement value of 19.30 dB.

Figure 6.6 shows the hard application of threshold was the most effective and produced an average increase in SNR of 7.24 dB at the 9th level of decomposition. This was supported by all the individual bearing signal groups except for the inner race bearings where the soft application of threshold was the best performer.

Figure 6.7 shows the best performing noise scale estimation was One and produced an average SNR increase of 15.31 dB at the 9th level of decomposition. Figures 6.19, 6.31, 6.43 and 6.55 support this as they all show One as the best noise estimation and also show the peak at the 9th level of wavelet decomposition. As expected the rolling element SNR improvement is high at 26.84 dB.

Figure 6.8 shows the comparison between all the Daubechies wavelets for envelope analysis. Daubechies wavelet DB19 was the best performing wavelet on average. The Daubechies wavelet DB19 produced an average increase in SNR of 11.23 dB at the 9th level of decomposition. The results for the envelope analysis of the normal bearings also selected Daubechies wavelet DB19 as the best performer with a SNR gain of 15.15 dB, as shown in Figure 6.20. The inner race faults (Figure 6.44) and the rolling element faults (Figure 6.32) signals were best improved using the Daubechies wavelet DB17 and the outer race faults best responded to the use of DB18. In all cases the 9th level of decomposition

provided the best SNR improvement from the wavelet de-noising process.

7.3.3 Cepstrum Analysis

The cepstrum results were checked for the attenuation issue identified in the Fourier analysis and envelope analysis. The issue was not found to affect the ceptrum analysis.

The combined results of the envelope analysis of short time signals acquired from various testbed arrangements are shown in Figures 6.9, 6.10, 6.11 and 6.12.

Figure 6.9 shows the most effective de-noising threshold was the universal threshold (SqTwoLog). It produced an average SNR gain of 3.33 dB at the 6th level of decomposition. Figures 6.45 and 6.57 show the universal threshold as the best option, providing peak SNR improvements at the 6th level of decomposition. Figure 6.21 shows the best threshold for the normal bearing signal was Minimaxi threshold and the peak occurred at the 8th level of decomposition.

Figure 6.10 shows the hard application of threshold was the most effective and produced an average increase in SNR of 2.16 dB at the 6th level of decomposition. The outer race (Figure 6.58) and normal bearing (Figure 6.22) signals also support this however the hard application of threshold for the normal bearing has the peak SNR gain at the 8th level of decomposition. Although the inner race and rolling element signals select the soft application of threshold they both have the peak SNR gain at the 6th level of decomposition.

Figure 6.11 shows the best performing noise scale estimation was One and produced and average increase in SNR of 4.04 dB at the 6th level of decomposition. Figures 6.47 and 6.35 also show noise scale estimation One as the best performer and both have peak SNR gains at the 6th level of decomposition. The outer race (Figure 6.59) and normal bearing (Figure 6.23) signals have the Mln noise scale estimation as the optimum parameter.

Figure 6.12 shows the comparison between all the Daubechies wavelets foe envelope analysis. Daubechies wavelet DB12 was the best performing wavelet on average. The Daubechies wavelet DB12 produced an average increase in SNR of 3.79 dB at the 6th level of decomposition. The inner race signals (Figure 6.48) and the rolling element signals (Figure 6.36) both had Daubechies wavelet DB12 as the best performing wavelet

and each had peak SNR gains at the 6th level of decomposition. Normal bearing signals (Figure 6.24) had DB12 as the best performing wavelet whereas the outer race signals (Figure 6.60) had DB18 as the best.

7.3.4 Conclusion

Fourier analysis of the wavelet de-noised short time signals resulted in negligible gains in SNR. The results indicated that varying the de-noising parameters of threshold, threshold application, and noise scale estimation have little effect on the improvement of SNR when analysing the short time signals using Fourier analysis. Figure 6.4 did show that certain wavelets could produce small gains in SNR and the best performing wavelet was DB19 with five levels of wavelet decomposition.

Envelope analysis and wavelet de-noising resulted in strong gains in SNR in the order of 7-13 dB. Although the higher levels of wavelet decomposition have produced erroneous results for the Fourier analysis, the gains presented for envelope analysis have been verified as real and the 9th level of decomposition was the most successful for envelope analysis.

Cepstrum analysis of the wavelet de-noised signal produced good gains in SNR when analysing the short time signals and was successful across the range of decomposition levels. The de-noising process improved the SNR of the signal by allowing cepstrum analysis to more easily extract the periodicities and produce stronger quefrency components.

7.4 Long Time Signals

The long time signals consisted of vibration data acquired from a single bearing over a period of 32 days of operation. Data files were acquired every 10 minutes over the 32 day period, however only one file from each day was used in this body of work.

7.4.1 Issue with higher levels of wavelet decomposition

Section 7.3.1 above identified an issue when using wavelet decomposition levels above level 5. The issue largely affected the Fourier analysis of short time signals but also had a

minor impact on envelope analysis. When processing the results of the long time signals a check was undertaken to confirm whether the above mentioned attenuation issue was again present in these results. The check consisted of reprocessing the long time signals and observing the frequency spectrum output when wavelet de-noising the signal at the higher levels of wavelet decomposition. When the issue was present, attenuation of the desired frequency components was clearly observed.

For the Fourier analysis results the attenuation issue was indeed present at wavelet decomposition levels above level 5. The typical sharp drop in SNR at decomposition levels six and seven were observed as shown in the Daubechies family comparison of Figure 7.10. However in this set of signals the issues did not produce large, unrealistic SNR gains at higher levels of decomposition, which was observed in Section 7.3.1. In most cases the SNR was largely reduced however there were some circumstances where small SNR gains were observed. These gains were not from improvements in the SNR from wavelet de-noising but instead were the result of attenuating most of the frequency components and grouping the remaining components around the cage frequency, typically at 10-15 Hz. For this reason the analysis of the results from Fourier analysis of the long time signals excluded SNR changes for wavelet decomposition levels above level 5.

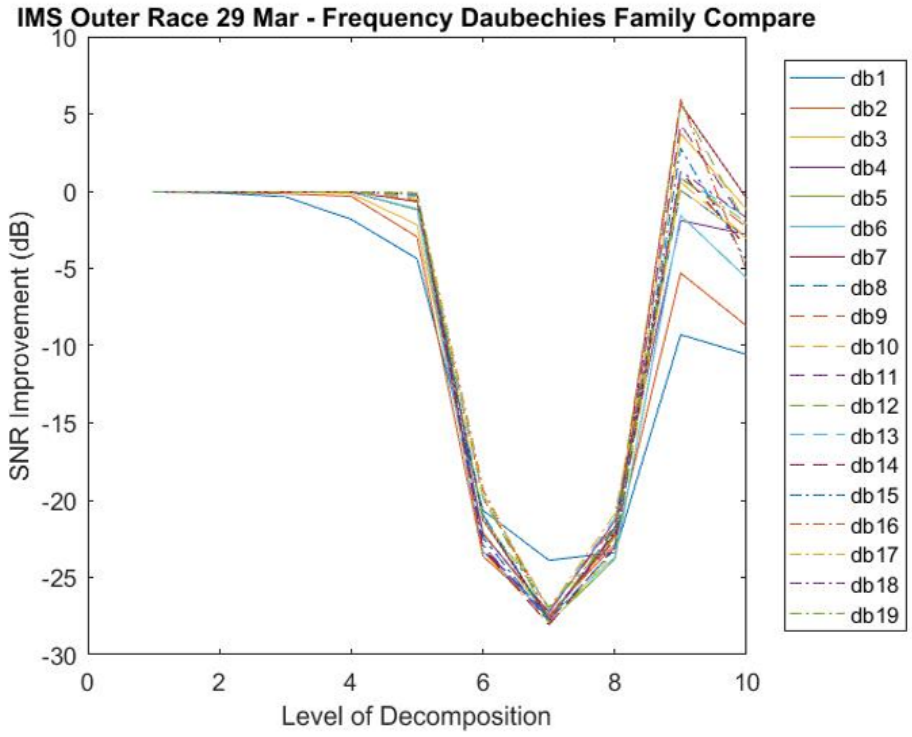

Figure 7.10: Daubechies wavelet family from 29 Mar showing the dip in SNR at levels 6, 7 and 8.

For the envelope analysis results the attenuation issue appeared in all file analysed. In Section 7.3.2 it was explained that this effect was present only for certain combinations of wavelet de-noising parameters and only affected certain bearing types. For the long time signals the attenuation issue was present for all decomposition levels above level 5 and occurred in every long time signal analysed. For this reason the analysis of the results from envelope analysis of the long time signals excluded SNR changes for wavelet decomposition levels above level 5.

The cesptrum analysis results were checked for the same issue by reprocessing the long time files and observing the quefrency domain produced for the higher levels of wavelet decomposition. No form of attenuation appeared in the cepstrum results and the full range of wavelet decomposition levels was used in the analysis.

7.4.2 Fourier Analysis

The results of the Fourier analysis of the simulation signals are shown in Tables 6.13, 6.14, 6.15, and 6.16. The results in the tables show that no significant gains in SNR resulted from wavelet de-noising the time domain signals and then processing using Fourier analysis. The largest SNR gain from the group of 32 signals was 0.89 dB however, in about 50% of the results the change in SNR was a reduction in signal quality.

Table 6.13 shows the threshold rigorous SURE (RigrSURE) was commonly the most effective thresholding method although it rarely improved the signal quality, it just reduced the SNR by the least amount compared to the other thresholds. Minimaxi and universal (SqTwoLog) thresholds also featured as the most effective for some signal files but again, neither threshold produced any significant increase in SNR. The 1st level of wavelet decomposition was the most effective for the thresholds in all but two signal files.

Table 6.14 indicates a soft application of the thresholds is the most commonly effective technique although no significant SNR gains were achieved. The 1st level of decomposition was the most commonly selected level for the signals.

Table 6.15 shows the best noise scale estimation method was One which outperformed the other methods in 19 of the 32 signals. The One noise scale estimation method was more suited to the latter half of the signal being analysed, presumably as the bearing

fault increased in severity. The best noise scale estimations occurred mostly at the 1st level of wavelet decomposition.

Table 6.16 shows the results for Daubechies wavelet family comparison. Three groups of best performing Daubechies wavelets appear in the results. The first group is around the Daubechies wavelets DB3 to DB9 and consist of 20 of the best performing wavelets. The second group is from DB15 to DB19 and consists of seven wavelets. The third group is made up of the DB10 to DB14 wavelets and has five of the best performing wavelets. The 4th level of wavelet decomposition was most commonly the best level selected.

7.4.3 Envelope Analysis

The results of the envelope analysis of the simulation signals are shown in Tables 6.17, 6.18, 6.19, and 6.20. These tables show only very minor gains in SNR resulting from wavelet de-noising a time domain signal and then processing using envelope analysis. It is worth noting however that unlike the Fourier analysis results, there were no reductions in the SNR when combine envelope analysis with wavelet de-noising.

Table 6.17 no significant SNR gains were made by varying the type of threshold when de-noising the signal. The table shows the most effective threshold was the universal threshold (SqTwoLog) and the best level of wavelet decomposition was the 3rd level followed closely by the 1st level.

Table 6.18 shows the hard application of the thresholds were the best performing method. Hard application featured as the best 23 of the possible 32 signals analysed. When comparing the soft or hard applications the 3rd level of wavelet decomposition was the best performing level across the range of 32 files. It is worth noting that the average SNR improvement in this comparison was only 0.03 dB.

Table 6.19 shows the results of the noise scale estimation comparison and it is clear that One noise scale estimation is the best performing method. The 3rd level of decomposition was the most common method selected.

Table 6.20 shows the best Daubechies wavelets were grouped around the DB1 to DB5 range. 25 of the best performing wavelets were in this group and DB1 featured as the best in 13 of the signals analysed. For the Daubechies comparison, the 5th level of wavelet

decomposition was the best level.

7.4.4 Cepstrum Analysis

The results of the envelope analysis of the simulation signals are shown in Tables 6.21, 6.22, 6.23, and 6.24.

Table 6.21 shows the universal threshold (SqTwoLog) method is clearly the best technique when wavelet de-noising the signal and then processing using cepstrum analysis. The universal threshold best performed at the 6th level of wavelet decomposition. The average SNR improvement over the 32 files was 26.22 dB.

Table **??** shows the hard application of the threshold is the best technique when de-noising the signals. Again, the 6th level of decomposition was the best performing level for the wavelet transform. The average SNR improvement over the 32 files was 18.33 dB.

Table 6.23 shows the best noise scale estimation is One when de-noising the long time signals and processing using cepstrum analysis. The optimum level of decomposition was level six and produced an average SNR gain of 36.05 dB.

Table **??** shows two main groups of Daubechies wavelets were the best performing wavelets. One group was centred around the Daubechies wavelets DB17 to DB19 and featured as the best in 13 of the signals. The other main group was centred around the DB10 to DB15 range and also featured 13 times as the best wavelets. The most common wavelet selected as the best was the Daubechies DB17 wavelet.

7.4.5 Conclusion

Fourier analysis of the wavelet de-noised long time signals showed the de-noising process actually reduced the SNR in most of the files analysed. For the comparisons of thresholds, threshold application, and noise scale estimation the 1st level of wavelet decomposition was commonly selected as the best option. Higher levels of decomposition in these comparisons further reduced the SNR. There was no clear standout for best Daubechies wavelet either. Many different wavelets featured as the best option for a particular file analysed but any SNR changes resulting were minor at best and negative at worst.

The performance by envelope analysis was only marginally better than Fourier analysis when analysing the long time signals. The exception here is that there were no reductions in SNR when envelope analysing the signals. Very small SNR gains were achieved using particular Daubechies wavelets however the wavelet de-noising parameters of threshold, threshold application, and noise scale estimation had little effect on changing the SNR results.

In contrast to the Fourier and envelope analysis the cepstrum analysis seemed to perform well when analysing the long time signals. Each category of comparison had clear standout parameters that performed the best at de-noising the signals and large gains in SNR were realised.

7.5 Results Comparison

The following tables display the optimum wavelet de-noising parameters for each CFF extraction method and allow comparison between the extraction methods and also between the three signal groups.

Table 7.1 shows the optimum parameters chosen from the simulation signals results. Table 7.2 shows the optimum parameters chosen from the short time signals results. Table 7.3 shows the optimum parameters chosen from the long time signals results.

Overall the CFF extraction method that responded best to wavelet de-noising was cepstrum analysis. In all three groups of signals (simulation, short time and long time signals) cepstrum analysis showed increases in SNR across the full range of signals analysed. The SNR improvement for all cepstrum analysed signals were consistent and the same threshold, threshold application, and noise scale estimation parameters were found to be the most successful across the three signal groups. When analysing the real vibration signals the 6th level of decomposition featured as the best level for improving the SNR.

Envelope analysis was the second best extraction method to respond to wavelet de-noising. Envelope analysis of the simulation signals and the short time signals showed good SNR gains resulting from wavelet de-noising however the de-noising process was not effective on the long time signals. Envelope analysis responded well to the same de-noising parameters as cepstrum analysis and shared similar optimum Daubechies wavelets and level of wavelet

decomposition for the simulation and short time signals.

The Fourier analysis method responded poorly to wavelet de-noising. Little or no SNR gains were achieved in all three signal groups and in most cases the SNR was reduced from the de-noising process. The optimum wavelet de-noising parameters were inconsistent for Fourier analysis across the three signal groups however the 1st level of decomposition was constantly selected as the best level.

Table 7.1: The best performing wavelet de-noising parameters for the simulation signals.

CFF Extraction Method	Threshold	Threshold Application	Noise Scale Estimation	Daubechies Wavelet	Level of Decomposition
Fourier	RigrSURE	Soft	Mln and Sln	DB11	1
Envelope	SqTwoLog	Hard	One	DB18	7
Cepstrum	SqTwoLog	Hard	One	DB4	2

Table 7.2: The best performing wavelet de-noising parameters for the short time signals.

CFF Extraction Method	Threshold	Threshold Application	Noise Scale Estimation	Daubechies Wavelet	Level of Decomposition
Fourier	SqTwoLog	Soft	One	DB19	1
Envelope	SqTwoLog	Hard	One	DB19	9
Cepstrum	SqTwoLog	Hard	One	DB2	6

Table 7.3: The best performing wavelet de-noising parameters for the long time signals.

CFF Extraction Method	Threshold	Threshold Application	Noise Scale Estimation	Daubechies Wavelet	Level of Decomposition
Fourier	RigrSURE	Soft	One	DB3	1
Envelope	SqTwoLog	Hard	One	DB1	4
Cepstrum	SqTwoLog	Hard	One	DB17	6

Chapter 8

Conclusions and Further Work

8.1 Chapter Overview

The broad aim of this project was to provide earlier detection of bearing faults by improving the SNR of a fault signal. The specific aim was to assess the large range of wavelet de-noising parameter combinations and determine which parameters were best suited to de-noising bearing vibration signals. My literature review revealed that similar but smaller studies had been undertaken in the past but these had only analysed simulation signals or a small range of testbed vibration signals. A further aim for this project then became analysing and testing a wider range of signals encompassing simulation and real vibration signals.

On this basis I set out to create a MATLAB^{TM} program that could analyse a wide range of vibration signals and test a wide range of wavelet de-noising parameters. This objective was achieved and the program I developed can take simulation or real vibration signals as inputs and iteratively test the performance of a range of wavelet de-noising parameter combinations. Over 1.5 million parameter combinations were tested across 111 vibration signal files.

The program I created uses three different methods for extracting the bearing CFFs. Cepstrum analysis was one of these methods and combined with wavelet de-noising I was able to greatly increase the SNR of the CFF across the three groups of signals (simulation, short time and long time signals). These increases in SNR however do not necessarily

equate to an earlier detection of bearing faults, which was one of the project aims. I had hoped to do more analysis with the long time signals in order to try an confirm if the SNR improvements would lead to an earlier detection of the fault. The long time signals were a collection of vibration signals acquired from a single bearing over a period of many days. It would have been feasible to cepstrum analyse the full range of long time signals and identify the moment in time when the bearing fault magnitude became sufficiently strong to be detected and identified as a fault. The long time signals could have then been wavelet de-noised and the process run again to determine if the fault could have been identified earlier in the series of vibration files. In the context of providing earlier detection of bearing faults the project has not achieved its aim however the strong increases in SNR are a promising start.

8.2 Further Work

In this body of work a method for estimating the SNR from within the frequency domain was adapted to measure signal improvement resulting from the wavelet de-noising process. The method used essentially sums the power of frequency bands surrounding the four CFFs and assumes these to be the desired signal. The power of remaining frequency components is assumed to be noise. This method performed satisfactorily in most cases of analysis however a flaw was discovered in the method when the signal became attenuated and only left low frequency components remaining. The low frequency components were grouped around one of the lower CFF providing strong signal energy at this frequency band and increasing the SNR to unrealistic levels. The process of estimating SNR from within the frequency domain could be explored further to avoid this discovered flaw and improve the SNR reporting. Further refining a method of SNR estimation within the frequency domain could have many applications not only for bearing vibration analysis but other areas of signal processing.

I think the most obvious direction for further work in this field is the analysis of a wider range of vibration signals, particularly signals acquired from the same bearing but at different stages of wear. This work analysed a large range of short time signals where bearings had been seeded with mechanical faults, producing strong frequency content at the correct CFFs. But for this work to have more relevance to the real world of condition monitoring of bearing vibrations the wavelet de-noising parameters need to be explored

across a range of vibration signals that don't have such strong frequency content at the correct CFFs. Quite a few bearing vibration datasets provide signals that have been acquired when the bearing was brand new through to when it was at the end of its life cycle. These are the types of signals that need to be analysed to determine how the wavelet de-noising parameters can best serve the condition monitoring process throughout the life of the bearing.

References

Abbasion, S., Rafsanjani, A., Farshidianfar, A. & Irani, N. (2007), 'Rolling Element Bearing Multi-fault Classification based on the Wavelet Denoising and Support Vector Machine', *Mechanical Systems and Signal Processing* **21**, 2933–2945.

Akturk, N. & Karacay, T. (2009), 'Experimental Diagnostics of Ball Bearings Using Statistical and Spectral Methods', *Tribology International* **42**, 836–843.

Al-Badour, F., Cheded, L. & Sunar, M. (2010), 'Non-stationary Vibrational Signal Analysis of Rotating Machinery via Time-Frequency and Wavelet Techniques', *10th International Conference on Information Science, Signal Processing and their Applicationas* pp. 21–24.

Al-Balushi, K. R. & Samanta, B. (2003), 'Artifical Neural Network Based Fault Diagnostics of Rolling Element Bearings Using Time Domain Features', *Mechanical Systems and Signal Processing* **17**(2), 317–328.

Antoniadis, I. A. & Nikolaou, N. G. (2002), 'Rolling Element Bearing Fault Diagnosis Using Wavelet Packets', *NDT and E International* **35**, 197–205.

Arniaz, A., Bediaga, I., Mendizabal, X. & Muñoa, J. (2013), 'Ball Bearing Damage Detection Using Traditional Signal Processing Algorithms', *IEEE Instrumentation and Measurement Magazine* pp. 20–25.

Bechhoefer, E. (2012), 'Condition Based Maintenance Fault Database for Testing of Diagnostic and Prognostics Algorithms', `http://www.mfpt.org/FaultData/FaultData.htm`. [Online; accessed October-2015].

Bogert, B. P., Healy, M. J. R. & Tukey, J. W. (1963), 'The Frequency Analysis of Time Series of Echoes: Cepstrum, Psuedo-autocovariance Cros-cepstrum and Saphe Cracking', *Proceedings of the Symposium on Time Series Analysis* pp. 209–243.

Bolaers, F., Dron, J. P. & Rasolofondraibe, L. (2004), 'Improvement of the Sensitivity of the Scalar Indicators (crest factor, kurtosis) using a De-noising Method by Spectral Subtraction: Application to the Detection of Defects in Ball Bearings', *Journal of Sound and Vibration* **270**, 61–73.

Cai, C. & Harrington, P. (1998), 'Different Discrete Wavelet Transforms Applied to Denoising Analytical Data', *Journal of Chemical Information and Computer Science* **38**(6), 1161–1170.

Case Western Reserve University (2015), 'Case Western Reserve University Bearing Data Center', `http://csegroups.case.edu/bearingdatacenter.htm`. [Online; accessed October-2015].

Center for Intelligent Maintenance Systems (2015), 'Bearing Dataset', `http://www.imscenter.net/`. [Online; accessed October-2015].

Chandra, N. H. & Sekhar, A. S. (2016), 'Fault Detection in Rotor Bearings Systems Using Time Frequency Techniques', *Mechanical Systems and Signal Processing* **72-73**, 105–133.

Choi, Y., Kim, Y. & Park, C. (2013), 'Early Fault Detection in Automotive Ball Bearings using the Minimum Variance Cepstrum', *Mechanical Systems and Signal Processing* **38**, 534–548.

Choudry, A. & Tandon, N. (1999), 'A Review of Vibration and Acoustic Measurement Methods for the Detection of Defects in Rolling Element Bearings', *Tribology International* **32**, 469–480.

data-acoustics (2015), 'Acoustics and Vibration Database', `http://data-acoustics.com/measurements/bearing-faults/htm`. [Online; accessed October-2015].

Donoho, D. (1995), 'De-Noising by Soft-Thresholding', *IEEE Transactions on Information Theory* **41**(3), 613–627.

Donoho, D. & Johnstone, I. (1994), 'Ideal Spatial Adaptation by Wavelet Shrinkage', *Biometrika* **81**(3), 425–455.

Donoho, D. & Johnstone, I. (1998), 'Minimax Estimation via Wavelet Shrinkage', *The Annals of Statistics* **26**(3), 879–921.

Dowling, M. J. (1993), 'Application of Non-stationary Analysis to Machinery Monitoring', *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993* **1**, 59–62.

el Badaoui, M., Daniére, J. & Guillet, F. (2004), 'New Applications of the Cepstrum to Gear Signals, Including Definition of a Robust Fault Indicator', *Mechanical Systems and Signal Processing* **18**, 1031–1046.

Ergen, B. (2012), Signal and image denoising using wavelet transform, *in* 'Advances in Wavelet Theroy and Their Applications in Engineering, Physics and Technology', InTech.

Forbes, G. (2012), 'Inner and Outer Race Bearing Fault Vibration Measurements', `http://data-acoustics.com/measurements/bearing-faults/bearing-1/`. [Online; accessed October-2015].

Fray, C., Pachaud, C. & Salvetat, R. (1997), 'Crest Factor and Kurtosis Contributions to Identify Defects Inducing Periodical Impulsive Forces', *Mechanical Systems and Signal Processing* **11**(6), 903–916.

Gade, S. & Gram-Hansen, K. (1996), 'Non-stationary Signal Analysis using Wavelet Transform, Short-time Fourier Transform and Wigner-Ville Distribution', *Technical Review - Brüel and Kjaer* (2).

Goyal, V. K., Kovačević, J. & Vertterli, M. (2013), *Fourier and Wavelet Signal Processing*.

Gustafsson, O. & Tallian, T. (1962), 'Detection of Damage in Assembled Rolling Element Bearings', *ASLE Transactions* **5**(1), 197–209.

Habetler, T. G., Harley, R. G. & Stack, J. R. (2004), 'An Amplitude Modulation Detector for Fault Diagnosis in Rolling Element Bearings', *IEEE Transactions on INdustrial Electronics* **51**(5), 1097–1102.

Harris, T. A. (1966), *Rolling Bearing Analysis*, Joh Wiley and Sons, New York.

Kalista, K. & Liska, J. (2015), 'Time-frequency Methods for Signal Analysis in Wind Turbines', *Journal of Physics: Conference Series* **659**.

Kasashima, N., Mori, K., Ueno, Y. & Yoshioka, T. (1996), 'Prediction of Spalling on a Ball Bearing by Applying the Discrete Wavelet Transform to Vibration Signals', *Wear* pp. 162–168.

Kiral, Z. & Karagülle, H. (2003), 'Simulation and Analysis of Vibration Signals Generated by Rolling Element Bearing with Defects', *Tribolgy International* **136**, 667–678.

Kostopouolos, V. & Loutas, T. (2012), Utilising the wavelet transform in condition based maintenance: A review with applications, *in* D. D. Baleanu, ed., 'Advances in Wavelet Theory and their Applications in Engineering, Physics and Technology', Intech.

Lin, J. (2001), 'Feature Extraction of Machine Sound Using Wavelet and its Application in Fault Diagnosis', *NDT and E International* **34**, 25–30.

Lin, J. & Qui, L. (2000), 'Feature Extraction Based on Morlet Wavelet and its Application for Mechanical Fault Diagnosis', *Journal of Sound and Vibration* **234**(1), 135–148.

Lin, J., Zuo, M. & Fyfe, K. (2004), 'Mechanical Fault Detection Based on the Wavelet De-Noising Technique', *Journal of Vibration and Acoustics* **126**, 9–16.

Luo, G. & Zhang, D. (2012), Wavelet denoising, *in* 'Advances in Wavelet Theroy and Their Applications in Engineering, Physics and Technology', InTech.

Lyons, R. (2011), *Understanding Digital Signal Processing*, Prentice Hall.

Machinery Failure Prevention Technology (2015), 'Condition Based Maintenance Fault Database for Testing of Diagnostic and Prognostics Algorithms', `http://www.mfpt.org/FaultData/FaultData.htm`. [Online; accessed October-2015].

Mallat, S. (1998), *A Wavelet Tour of Signal Processing*, Academic Press.

Mathew, J., Patil, M. S. & RajendraKumar, P. K. (2008), 'Bearing Signature Analysis as a Medium for Fault Detection: A Review', *Journal of Tribology* **130**.

McInerny, S. & Dai, Y. (2003), 'Basic Vibration Signal Processing for Bearing Fault Detection', *IEEE Transactions on Education* **46**(1), 149–156.

Mechefske, C. K. (2005), Machine condition monitoring and fault diagnostics, *in* C. W. de Silva, ed., 'Vibration and Shock Handbook', Taylor and Francis Group, LLC.

Misiti, M., Misiti, Y., Oppenheim, G. & Poggi, J. (2016), *Wavelet Toolbox Users Guide*, Mathworks.

Mohanty, A. R., Prabhakar, S. & Sekhar, A. S. (2002), 'Application of Discrete Wavelet Transform for Detection of Ball Bearing Race Faults', *Tribology International* **35**, 793–800.

Momono, T. & Noda, B. (1999), 'Sound and Vibration in Rolling Bearings', *Motion and Control - NSK* (6), 29–37.

National Aeronautics and Space Administration (2015), 'Prognostics Center of Excellence', `https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/`. [Online; accessed October-2015].

Qiu, H., Lee, J., Lin, J. & Yu, G. (2006), 'Wavelet Filter-based Weak Signature Detection Method and its Application on Rolling Element Bearing Prognostics', *Journal of Sound and Vibration* **289**, 1066–1090.

Staszewski, W. (1998), 'Wavelet Based Compression and Feature Selection for Vibration Analysis', *Journal of Sound and Vibration* **211**(5), 735–760.

Stein, C. (1981), 'Estimation of the Mean of a Multivariate Normal Distribution', *The Annals of Statistics* **9**(6), 1135–1151.

Zhang, J. & Liu, S. (2010), 'Study on the Comparison of Several Denoising Theories and Effects About Vibration Signal', *2010 International Symposium on Computational Intelligence and Design* pp. 136–139.

Appendix A

Project Specification

ENG 4111/2 Research Project

Project Specification

For:	**James Steele**
Topic:	Signal Processing Techniques for Machine Condition Monitoring
Supervisor:	J. Leis
Sponsorship:	Faculty of Health, Engineering & Sciences
Project Aim:	To provide earlier prediction of bearing failure by extracting very weak bearing fault signals from vibration signals dominated by noise.

Program:

1. Review the literature on DSP for machine condition monitoring, summarise earlier developments and current research trends.
2. Critically examine machine condition monitoring techniques found during the literature search and document their applicability to various scenarios.
3. Seek out existing machine fault databases. If available data is not suitable then design and manufacture a testbed for creating vibration signal data.
4. Obtain MATLAB^{TM} data access functions where required. If data access functions are not available then write appropriate code libraries.
5. Critically examine wavelet theory and other methods for improving the timeliness and accuracy of detection of machine fault signals.
6. Design an improved algorithm that provides earlier detection of machine fault signals.
7. Test and evaluate the algorithm using artificial and real machine data signals.

As time and resources permit:

1. Investigate the relationship between the SNR and diagnosis methods.
2. Model the effect the phase difference has as each sensor records a vibration signal at slightly different moments.

3. Investigate the application of the above algorithms for use with alternative machine fault signals.

Agreed:

Student Name:	James Steele
Date:	1 March 2016
Supervisor Name:	John Leis
Date:	1 March 2016

Appendix B

MATLAB^{TM} programs written for this dissertation

B.1 The program data_file_assess_vthree.m

Listing B.1: The main program.

```
%*************************************************************************
% File Name: data_file_assess_vthree
% Author: James Steele
% First Written: 12 August 2016
%
% Purpose: The program takes vibration signals as input files and processes
% to extract frequency content. The SNR is calculated from thee
% frequency content. Next the input file is wavelet de-noised, thee
% frequency content extracted again and the new SNR is calculated. One
% parameter of the wavelet de-noising process is changed and the frequency
% content extracted again.The change in SNR is stored in a matrix and
% saved as an output file.
%
%*************************************************************************
clear all;
close all;
%% Load the data file
file1 = 'NASA_OR_19Mar_ch3.mat';
file2 = 'NASA_OR_20Mar_ch3.mat';
file3 = 'NASA_OR_21Mar_ch3.mat';
file4 = 'NASA_OR_22Mar_ch3.mat';
file5 = 'NASA_OR_23Mar_ch3.mat';
file6 = 'NASA_OR_24Mar_ch3.mat';
file7 = 'NASA_OR_25Mar_ch3.mat';
file8 = 'NASA_OR_26Mar_ch3.mat';
file9 = 'NASA_OR_27Mar_ch3.mat';
file10 = 'NASA_OR_28Mar_ch3.mat';
file11 = 'NASA_OR_29Mar_ch3.mat';
file12 = 'NASA_OR_30Mar_ch3.mat';
file13 = 'NASA_OR_31Mar_ch3.mat';
file14 = 'NASA_OR_01Apr_ch3.mat';
file15 = 'NASA_OR_02Apr_ch3.mat';
file16 = 'NASA_OR_03Apr_ch3.mat';
file17 = 'NASA_OR_04Apr_ch3.mat';
file = [file1; file2; file3; file4; file5; file6; file7;...
    file8; file9; file10; file11; file12; file13; file14; file15;...
    file16; file17];

% get the size of file
[in_num, in_leng] = size(file);

output1 = 'NASA_OR_19Mar_results.mat';
output2 = 'NASA_OR_20Mar_results.mat';
output3 = 'NASA_OR_21Mar_results.mat';
output4 = 'NASA_OR_22Mar_results.mat';
output5 = 'NASA_OR_23Mar_results.mat';
```

```
output6 = 'NASA_OR_24Mar_results.mat';
output7 = 'NASA_OR_25Mar_results.mat';
output8 = 'NASA_OR_26Mar_results.mat';
output9 = 'NASA_OR_27Mar_results.mat';
output10 = 'NASA_OR_28Mar_results.mat';
output11 = 'NASA_OR_29Mar_results.mat';
output12 = 'NASA_OR_30Mar_results.mat';
output13 = 'NASA_OR_31Mar_results.mat';
output14 = 'NASA_OR_01Apr_results.mat';
output15 = 'NASA_OR_02Apr_results.mat';
output16 = 'NASA_OR_03Apr_results.mat';
output17 = 'NASA_OR_04Apr_results.mat';

output = [output1; output2; output3; output4; output5; output6;...
    output7; output8; output9; output10; output11; output12;...
    output13; output14; output15; output16; output17];

% get the size of output
[out_num, out_leng] = size(output);

%% Set global variables
%     ***** Variables that apply to all 3 methods *****

% frame length of the window to be analysed. This sets the freq resolution
% for freq_spec and envelope based methods but also the window is also
% analysed by cepstrum
%frame_len = 65536;
%frame_len = 32768;
%frame_len = 16384;
frame_len = 8192;
%frame_len = 4096;
%frame_len = 2048;

% set the approximate desired quantity of frames
num_frames = 10;

% the correct sample frequency of the input files
%samp_freq = 12000;  %Case files
samp_freq = 20000;   % NASA testbed
%samp_freq = 48000;  %Case files
%samp_freq = 48828;  %MFPT files 7 outer and 7 inner
%samp_freq = 51200;   %inner_outer_race
%samp_freq = 97656;  %MFPT basleine

% the time of the frame_len
T=frame_len/samp_freq;

% calculate the frequency bin values
freq_bin_vals=(0:frame_len-1)/T;

%% Define all frequencies/quefrencies
% define the target frequencies to be observed or suppressed
```

```
%shaft_freq = 29.93;     %shaft speed will be suppressed
% _0
%OR = 107.31;
%IR = 162.09;
%BSF = 141.09;
%cage = 11.92;

% _1
%OR = 105.86;
%IR = 159.91;
%BSF = 139.19;
%cage = 11.76;

% _2
%OR = 104.43;
%IR = 157.74;
%BSF = 137.3;
%cage = 11.6;

% _3
%OR = 103.06;
%IR = 155.69;
%BSF = 135.51;
%cage = 11.45;

% inner outer race 29hz
%OR = 103.59;
%IR = 157.41;
%BSF = 67.28;
%cage = 11.51;

% MFPT
%OR = 81.13;
%IR = 118.88;
%BSF = 63.91;
%cage = 14.84;

% NASA
OR = 236.38;
IR = 293.96;
BSF = 139.9;
cage = 14.77;

% define the quefrency equivalents - in milliseconds
%shaft_quef = (1/shaft_freq)*1000;   %shaft speed will be suppressed
BPO_quef = (1/OR)*1000;
BPI_quef = (1/IR)*1000;
BSF_quef = (1/BSF)*1000;
cage_quef = (1/cage)*1000;

% define the acceptable bandwidth around the frequencies.
band_freq = 10;       %this creates a 20 Hz bandwith
```

```
band_peak = 2;      %this will meausure the 4 Hz around the peak component

% define the lower and upper limits of the acceptable bandwidths around the
% frequencie/quefrencies. These are the bandwiths that will be searched for
% the bearing frequencies.
OR_low = OR-band_freq;
OR_up = OR+band_freq;
IR_low = IR-band_freq;
IR_up = IR+band_freq;
BSF_low = BSF-band_freq;
BSF_up = BSF+band_freq;
cage_low = cage-band_freq;
cage_up = cage+band_freq;

% these set the lower/upper quefrency values (milliseconds) so it uses
% the upper/lower frequency values.
OR_quef_low = (1/OR_up)*1000;
OR_quef_up = (1/OR_low)*1000;
IR_quef_low = (1/IR_up)*1000;
IR_quef_up = (1/IR_low)*1000;
BSF_quef_low = (1/BSF_up)*1000;
BSF_quef_up = (1/BSF_low)*1000;
cage_quef_low = (1/cage_up)*1000;
cage_quef_up = (1/cage_low)*1000;

%     ***** Variables just for frequency spectrum *****

% the frequency band to be displayed. From lower to upper frequency in Hz
%upper_BW = 250;     % all data files except the NASA set
upper_BW = 300;      % The peak freq for the NASA set is 293Hz
lower_BW = 10;

% create the blank space for normal FFT spectrum
normal_FFT = zeros(1,frame_len);
normal_FFT_den = zeros(1, frame_len);

%     ***** Variables just for envelope analysis *****

% create the blank space for hilbert spectrum
hilbert_spec = zeros(1,frame_len);
hilbert_spec_den = zeros(1,frame_len);

%     ***** Variables just for cepstrum *****

% cepstrum frame length controls the length of the cepstrum x axis. A
% larger value increases the time (ms) being observed - ie observes lower
% frequencies
%cep_frame_len = 1024;
%cep_frame_len = 2048;
%cep_frame_len = 4096;  % for the CASE files
cep_frame_len = 8192;
%cep_frame_len = 16384; % for MFPT files with 48k samp freq
```

```
%cep_frame_len = 32768;  % for MFPT baseline files - 98k samp freq

% calculate the number of unique points. Only the first half of the values
% of the cepstrum are used, the second half of the values are the mirror of
% the first half
num_unique_pts = ceil((cep_frame_len+1)/2);

% calculate the quefrency vector (the x axis of the cepstrum). x1000
% because the quefrency will be in milliseconds.
data_q = ((1:num_unique_pts)/samp_freq)*1000;

% remove the cepstrum content less than 4 ms as this is frequency content
% above 250Hz. All files except the NASA dataset
%cep_low = 4;
% for the NASA data set, the min cepstrum value is 3 ms = 333Hz
cep_low = 3;
data_q(data_q < cep_low) = 0;
index_low = find(data_q,1);

% remove cepstrum content greater than 125 ms (8 Hz) as this is frequency
% content below the cage frequency (typically 11Hz min in all the data files
% being tested)
cep_up = 125;
data_q(data_q > cep_up) = 0;
index_up = find(data_q,1, 'last');

% set the new length of data_q with the 1st X ms removed
data_q = data_q(1, index_low:index_up);

% create empty c to store cepstrum values
c = zeros(1,length(data_q));
c_den = zeros(1, length(data_q));

%%     ***** set all the wavelet denoise variables in vectors *****
% the 4 threshold methods
thr1 = 'rigrsure';
thr2 = 'sqtwolog';
thr3 = 'heursure';
thr4 = 'minimaxi';
thresh = [thr1; thr2; thr3; thr4];

% 2 methods to apply threshold - soft or hard
sorh1 = 's';
sorh2 = 'h';
sorh = [sorh1; sorh2];

% the 3 types of noise
scal1 = 'one';
scal2 = 'sln';
scal3 = 'mln';
scal = [scal1; scal2; scal3];
```

```
% the first 19 types of daubachies wavelet
wav1 = 'db1';
wav2 = 'db2';
wav3 = 'db3';
wav4 = 'db4';
wav5 = 'db5';
wav6 = 'db6';
wav7 = 'db7';
wav8 = 'db8';
wav9 = 'db9';
wav10 = 'db10';
wav11 = 'db11';
wav12 = 'db12';
wav13 = 'db13';
wav14 = 'db14';
wav15 = 'db15';
wav16 = 'db16';
wav17 = 'db17';
wav18 = 'db18';
wav19 = 'db19';

% Matlab wont allow the 3 char and 4 char variables in the same vector
wav = [wav1; wav2; wav3; wav4; wav5; wav6; wav7; wav8; wav9];
wav2 = [wav10; wav11; wav12; wav13; wav14; wav15; wav16; wav17; wav18; wav19];

%% create the master results matrices for the 3 methods
% store the average change in SNR
master_results_env = zeros((length(wav)+length(wav2))*6, 40);

% store the average change in SNR
master_results_freq = zeros((length(wav)+length(wav2))*6, 40);

% store the average change in SNR
master_results_cep = zeros((length(wav)+length(wav2))*6, 40);

for d = 1:in_num  % loops as many times as there are input files
    % load the data and access the channel of data
    datastruct = load(file(d,1:in_leng));

    % use fieldnames() to extract the variable names
    fn = fieldnames(datastruct);

    % import the particular data series from the datastruct. The first
    % fieldname is the DE dataset
    %extra_step = datastruct.(fn{1});
    %data = extra_step.gs;
    data = datastruct.(fn{1});
    data = data';

    % based on the frame length, get the number frames possible
    max_frame_num = length(data) - frame_len;
```

```
    % calculate the offset based on the possible number of frames and the
    % desired number
    offset = floor(max_frame_num/num_frames);

    % do the Hilbert transform to the whole data vector
    analytic = hilbert(data);
    analytic = abs(analytic);

%% Process the original data file before denoising
% set the start points for the windows
win_start = 1;
win_end = frame_len;

% loop through num_frames times to get average frequency/quefrency content
for p = 1:num_frames

    % load the window with a frame of data for regular and Hilbert data
    window = data(win_start:win_end);
    window_analytic = analytic(win_start:win_end);

    % move the start and end points
    win_start = win_start + offset;
    win_end = win_end + offset;

    %% Envelope analysis
    % get the frequency spectrum from the Hilbert transformed data
    % (frame_len/2) corrects the amplitude of the FFT components
    hilbert_spec = hilbert_spec + abs(fft(window_analytic,frame_len))/...
        (frame_len/2);

    %% Normal frequency analysis
    normal_FFT = normal_FFT + abs(fft(window,frame_len))/(frame_len/2);

    %% cepstrum content extraction
    % power cepstrum
    cep = (abs(ifft(log((abs(fft(window))).^2)))).^2;

    % remove one side of the cepstrum because real signals have amplitude
    % symmetry. Also remove the first 2 elements because of large values
    % here masking other quefrencies
    c = c + cep(index_low:index_up);
end

%% Calculate the average and then power of the 3 methods
% averages the values
hilbert_spec = hilbert_spec./num_frames;
normal_FFT = normal_FFT./num_frames;
c = c./num_frames;

% convert to power
hilbert_spec = hilbert_spec.^2;
normal_FFT = normal_FFT.^2;
```

```
%% Reduce the spectrums down to the desired bandwidth
% envelope
% combine the hilbert and frequency bins into a single array
full_FFT_hilbert(1,:) = hilbert_spec(1,:);
full_FFT_hilbert(2,:) = freq_bin_vals(1,:);

% select the bandwidth for further analysis
selected_FFT_hilbert = freq_limit(full_FFT_hilbert, lower_BW, upper_BW);
selected_FFT_hilbert = selected_FFT_hilbert(:,2:end);

% normal frequency
% combine the normal spectrum and frequency bins into a 2 row array
freq_spec(1,:) = normal_FFT(1,:);
freq_spec(2,:) = freq_bin_vals(1,:);

% select the bandwidth for further analysis
selected_FFT = freq_limit(freq_spec, lower_BW, upper_BW);
selected_FFT = selected_FFT(:,2:end);

%% Get the strength of each fault freq/quef component
% envelope
env_OR = freq_energy_calc(selected_FFT_hilbert,OR_low,OR_up);
env_IR = freq_energy_calc(selected_FFT_hilbert, IR_low, IR_up);
env_BSF = freq_energy_calc(selected_FFT_hilbert, BSF_low, BSF_up);
env_cage = freq_energy_calc(selected_FFT_hilbert, cage_low, cage_up);

% normal frequency
freq_OR = freq_energy_calc(selected_FFT,OR_low,OR_up);
freq_IR = freq_energy_calc(selected_FFT, IR_low, IR_up);
freq_BSF = freq_energy_calc(selected_FFT, BSF_low, BSF_up);
freq_cage = freq_energy_calc(selected_FFT, cage_low, cage_up);

% cepstrum
cep_OR = cep_energy_calc(c, data_q, OR_quef_low, OR_quef_up);
cep_IR = cep_energy_calc(c, data_q, IR_quef_low, IR_quef_up);
cep_BSF = cep_energy_calc(c, data_q, BSF_quef_low, BSF_quef_up);
cep_cage = cep_energy_calc(c, data_q, cage_quef_low, cage_quef_up);

%% calculate the SNR for each method
% enveolpe
hilbert_CFF = env_OR + env_IR + env_BSF + env_cage;
hilbert_noise = sum(selected_FFT_hilbert(1,:)) - hilbert_CFF;
SNR_hilbert = 10.*log(hilbert_CFF/hilbert_noise);

% normal frequency
normal_freq_CFF = freq_OR + freq_IR + freq_BSF + freq_cage;
normal_freq_noise = sum(selected_FFT(1,:)) - normal_freq_CFF;
SNR_normal_freq = 10.*log(normal_freq_CFF/normal_freq_noise);

% cepstrum
cepstrum_CFF = cep_OR + cep_IR + cep_BSF + cep_cage;
```

```
cepstrum_noise = sum(c(1,:)) - cepstrum_CFF;
SNR_cepstrum = 10.*log(cepstrum_CFF/cepstrum_noise);

% plots are for debugging and observing freq/quef content
%figure(1);
%plot(selected_FFT_hilbert(2,:), selected_FFT_hilbert(1,:));
%title('Envelope analysis');
%xlabel('Frequency (Hz)');
%ylabel('Magnitude');

%figure(2);
%plot(selected_FFT(2,:), selected_FFT(1,:));
%title('Fourier analysis');
%xlabel('Frequency (Hz)');
%ylabel('Magnitude');

%figure(3);
%plot(data_q, c);
%title('Cepstrum analysis');
%xlabel('Quefrency (ms)');
%ylabel('Magnitude');

%% Denoise the data then process
% loops through every combination of wavelet and measures the SNR
% improvement
for i = 4:length(wav)          % sets the wavelet type

for j = 1:2                    % sets SORH
    for k = 1:3                % sets the noise scale
        for l = 1:4            % sets the thresholds
            for m = 1:10        % sets the level of decomposition

%% Denoise each window and window analytic
data_den = wden(data, thresh(l,1:8), sorh(j,1),...
    scal(k, 1:3), m, wav(i, 1:3));
data_an_den = wden(analytic, thresh(l,1:8), sorh(j,1),...
    scal(k, 1:3), m, wav(i, 1:3));

%% Clear the accumulating variables
hilbert_spec_den(1,:) = 0;
normal_FFT_den(1,:) = 0;
c_den(1,:) = 0;

%% Cycle through the data num_frames number of times
% reset the start points for the windows
win_start = 1;
win_end = frame_len;

for p = 1:num_frames
    % load the window with a frame of data for regular and Hilbert data
    win_den = data_den(win_start:win_end);
    win_an_den = data_an_den(win_start:win_end);
```

```
        % move the start and end points
        win_start = win_start + offset;
        win_end = win_end + offset;

        %% Envelope analysis on the denoised signal
        % get the frequency spectrum from the Hilbert transformed data
        hilbert_spec_den = hilbert_spec_den + abs(fft(win_an_den,frame_len))/...
            (frame_len/2);

        %% Normal frequency extraction on the denoised signal
        normal_FFT_den = normal_FFT_den + abs(fft(win_den,frame_len))/...
            (frame_len/2);

        %% cepstrum content extraction from the denoised signal
        % power cepstrum
        cep_den = (abs(ifft(log((abs(fft(win_den))).^2)))).^2;

        % remove one side of the cepstrum because real signals have amplitude
        % symmetry. Also remove the first few elements because of large values
        % here masking other quefrencies
        c_den = c_den + cep_den(index_low:index_up);

    end

    %% Calculate the average and then power of the 3 methods
    % averages the values
    hilbert_spec_den = hilbert_spec_den./num_frames;
    normal_FFT_den = normal_FFT_den./num_frames;
    c_den = c_den./num_frames;

    % convert to power
    hilbert_spec_den = hilbert_spec_den.^2;
    normal_FFT_den = normal_FFT_den.^2;

    %% Reduce the spectrums down to the desired bandwidth
    % envelope
    % combine the hilbert and frequency bins into a single array
    full_FFT_hilbert_den(1,:) = hilbert_spec_den(1,:);
    full_FFT_hilbert_den(2,:) = freq_bin_vals(1,:);

    % select the bandwidth for further analysis
    selected_FFT_hilbert_den = freq_limit(full_FFT_hilbert_den,...
            lower_BW, upper_BW);
    selected_FFT_hilbert_den = selected_FFT_hilbert_den(:,2:end);

    % normal frequency
    % combine the normal spectrum and frequency bins into a 2 row array
    freq_spec_den(1,:) = normal_FFT_den(1,:);
    freq_spec_den(2,:) = freq_bin_vals(1,:);

    % select the bandwidth for further analysis
```

```
selected_FFT_den = freq_limit(freq_spec_den, lower_BW, upper_BW);
selected_FFT_den = selected_FFT_den(:,2:end);

%% get the strength of each fault freq/quef component
% envelope denoised
env_OR_den = freq_energy_calc(selected_FFT_hilbert_den,OR_low,OR_up);
env_IR_den = freq_energy_calc(selected_FFT_hilbert_den, IR_low, IR_up);
env_BSF_den = freq_energy_calc(selected_FFT_hilbert_den, BSF_low, BSF_up);
env_cage_den = freq_energy_calc(selected_FFT_hilbert_den, cage_low, cage_up);

% normal frequency denoised
freq_OR_den = freq_energy_calc(selected_FFT_den,OR_low,OR_up);
freq_IR_den = freq_energy_calc(selected_FFT_den, IR_low, IR_up);
freq_BSF_den = freq_energy_calc(selected_FFT_den, BSF_low, BSF_up);
freq_cage_den = freq_energy_calc(selected_FFT_den, cage_low, cage_up);

% cepstrum denoised
cep_OR_den = cep_energy_calc(c_den, data_q, OR_quef_low, OR_quef_up);
cep_IR_den = cep_energy_calc(c_den, data_q, IR_quef_low, IR_quef_up);
cep_BSF_den = cep_energy_calc(c_den, data_q, BSF_quef_low, BSF_quef_up);
cep_cage_den = cep_energy_calc(c_den, data_q, cage_quef_low, cage_quef_up);

%% Calculate the SNR for each method after denoising
% enveolpe after denoising
hilbert_CFF_den = env_OR_den + env_IR_den + env_BSF_den + env_cage_den;
hilbert_noise_den = sum(selected_FFT_hilbert_den(1,:)) - hilbert_CFF_den;
SNR_hilbert_den = 10.*log(hilbert_CFF_den/hilbert_noise_den);

% normal frequency after denoising
normal_freq_CFF_den = freq_OR_den + freq_IR_den + freq_BSF_den +...
    freq_cage_den;
normal_freq_noise_den = sum(selected_FFT_den(1,:)) - normal_freq_CFF_den;
SNR_normal_freq_den = 10.*log(normal_freq_CFF_den/normal_freq_noise_den);

% cesptrum after denoising
cepstrum_CFF_den = cep_OR_den + cep_IR_den + cep_BSF_den + cep_cage_den;
cepstrum_noise_den = sum(c_den(1,:)) - cepstrum_CFF_den;
SNR_cepstrum_den = 10.*log(cepstrum_CFF_den/cepstrum_noise_den);

%% calculate the change in SNR from denoising
change_env = SNR_hilbert_den - SNR_hilbert;
change_freq = SNR_normal_freq_den - SNR_normal_freq;
change_cep = SNR_cepstrum_den - SNR_cepstrum;

%% store the change in SNR in the master results array
% create the row and col reference points
row_start = i*6-5;
row_offset = (j-1)*3 + k-1;
col_start = m*4-3;
col_offset = l-1;

master_results_env(row_start+row_offset, col_start+col_offset)...
```

```
    = change_env;
master_results_freq(row_start+row_offset, col_start+col_offset)...
    = change_freq;
master_results_cep(row_start+row_offset, col_start+col_offset)...
    = change_cep;

% more plots for debugging
%figure(4);
%plot(selected_FFT_hilbert_den(2,:), selected_FFT_hilbert_den(1,:));
%title('Envelope analysis denoised');

%figure(5);
%plot(selected_FFT_den(2,:), selected_FFT_den(1,:));
%title('Fourier analysis denoised - decomposition level 10');
%xlabel('Frequency (Hz)');
%ylabel('Magnitude');

%figure(6);
%plot(data_q, c_den);
%title('Cepstrum analysis denoised');

            end
        end
    end
end
end

%% Process the second half of the wavelets
for i = 1:length(wav2)        % sets the wavelet type

for j = 1:2                   % sets SORH
    for k = 1:3               % sets the noise scale
        for l = 1:4           % sets the thresholds
            for m = 1:10      % sets the level of decomposition

%% Denoise each window and window analytic
data_den = wden(data, thresh(l,1:8), sorh(j,1),...
    scal(k, 1:3), m, wav2(i, 1:4));
data_an_den = wden(analytic, thresh(l,1:8), sorh(j,1),...
    scal(k, 1:3), m, wav2(i, 1:4));

%% Clear the accumulating variables
hilbert_spec_den(1,:) = 0;
normal_FFT_den(1,:) = 0;
c_den(1,:) = 0;

%% Cycle through the data num_frames number of times
% reset the start points for the windows
win_start = 1;
win_end = frame_len;

for p = 1:num_frames
```

```
    % load the window with a frame of data for regular and Hilbert data
    win_den = data_den(win_start:win_end);
    win_an_den = data_an_den(win_start:win_end);

    % move the start and end points
    win_start = win_start + offset;
    win_end = win_end + offset;

    %% Envelope analysis on the denoised signal
    % get the frequency spectrum from the Hilbert transformed data
    hilbert_spec_den = hilbert_spec_den + abs(fft(win_an_den,frame_len))/...
        (frame_len/2);

    %% Normal frequency extraction on the denoised signal
    normal_FFT_den = normal_FFT_den + abs(fft(win_den,frame_len))/...
        (frame_len/2);

    %% cepstrum content extraction from the denoised signal
    % power cepstrum
    cep_den = (abs(ifft(log((abs(fft(win_den))).^2)))).^2;

    % remove one side of the cepstrum because real signals have amplitude
    % symmetry. Also remove the first 2 elements because of large values
    % here masking other quefrencies
    c_den = c_den + cep_den(index_low:index_up);

end

%% Calculate the average and then power of the 3 methods
% averages the values
hilbert_spec_den = hilbert_spec_den./num_frames;
normal_FFT_den = normal_FFT_den./num_frames;
c_den = c_den./num_frames;

% convert to power
hilbert_spec_den = hilbert_spec_den.^2;
normal_FFT_den = normal_FFT_den.^2;
%c_den = c_den.^2;

%% Reduce the spectrums down to the desired bandwidth
% envelope
% combine the hilbert and frequency bins into a single array
full_FFT_hilbert_den(1,:) = hilbert_spec_den(1,:);
full_FFT_hilbert_den(2,:) = freq_bin_vals(1,:);

% select the bandwidth for further analysis
selected_FFT_hilbert_den = freq_limit(full_FFT_hilbert_den,...
        lower_BW, upper_BW);
selected_FFT_hilbert_den = selected_FFT_hilbert_den(:,2:end);

% normal frequency
% combine the normal spectrum and frequency bins into a 2 row array
```

```
freq_spec_den(1,:) = normal_FFT_den(1,:);
freq_spec_den(2,:) = freq_bin_vals(1,:);

% select the bandwidth for further analysis
selected_FFT_den = freq_limit(freq_spec_den, lower_BW, upper_BW);
selected_FFT_den = selected_FFT_den(:,2:end);

%% get the strength of each fault freq/quef component
% envelope denoised
env_OR_den = freq_energy_calc(selected_FFT_hilbert_den,OR_low,OR_up);
env_IR_den = freq_energy_calc(selected_FFT_hilbert_den, IR_low, IR_up);
env_BSF_den = freq_energy_calc(selected_FFT_hilbert_den, BSF_low, BSF_up);
env_cage_den = freq_energy_calc(selected_FFT_hilbert_den, cage_low, cage_up);

% normal frequency denoised
freq_OR_den = freq_energy_calc(selected_FFT_den,OR_low,OR_up);
freq_IR_den = freq_energy_calc(selected_FFT_den, IR_low, IR_up);
freq_BSF_den = freq_energy_calc(selected_FFT_den, BSF_low, BSF_up);
freq_cage_den = freq_energy_calc(selected_FFT_den, cage_low, cage_up);

% cepstrum denoised
cep_OR_den = cep_energy_calc(c_den, data_q, OR_quef_low, OR_quef_up);
cep_IR_den = cep_energy_calc(c_den, data_q, IR_quef_low, IR_quef_up);
cep_BSF_den = cep_energy_calc(c_den, data_q, BSF_quef_low, BSF_quef_up);
cep_cage_den = cep_energy_calc(c_den, data_q, cage_quef_low, cage_quef_up);

%% Calculate the SNR for each method after denoising
% enveolpe after denoising
hilbert_CFF_den = env_OR_den + env_IR_den + env_BSF_den + env_cage_den;
hilbert_noise_den = sum(selected_FFT_hilbert_den(1,:)) - hilbert_CFF_den;
SNR_hilbert_den = 10.*log(hilbert_CFF_den/hilbert_noise_den);

% normal frequency after denoising
normal_freq_CFF_den = freq_OR_den + freq_IR_den + freq_BSF_den +...
    freq_cage_den;
normal_freq_noise_den = sum(selected_FFT_den(1,:)) - normal_freq_CFF_den;
SNR_normal_freq_den = 10.*log(normal_freq_CFF_den/normal_freq_noise_den);

% cesptrum after denoising
cepstrum_CFF_den = cep_OR_den + cep_IR_den + cep_BSF_den + cep_cage_den;
cepstrum_noise_den = sum(c_den(1,:)) - cepstrum_CFF_den;
SNR_cepstrum_den = 10.*log(cepstrum_CFF_den/cepstrum_noise_den);

%% calculate the change in SNR from denoising
change_env = SNR_hilbert_den - SNR_hilbert;
change_freq = SNR_normal_freq_den - SNR_normal_freq;
change_cep = SNR_cepstrum_den - SNR_cepstrum;

%% store the change in SNR in the master results array
% create the row and col reference points
row_start = i*6-5+54;
row_offset = (j-1)*3 + k-1;
```

```
col_start = m*4-3;
col_offset = l-1;

master_results_env(row_start+row_offset, col_start+col_offset)...
    = change_env;
master_results_freq(row_start+row_offset, col_start+col_offset)...
    = change_freq;
master_results_cep(row_start+row_offset, col_start+col_offset)...
    = change_cep;

% more plots for debuggin
%figure(4);
%plot(selected_FFT_hilbert_den(2,:), selected_FFT_hilbert_den(1,:));
%title('Envelope analysis denoised');

%figure(5);
%plot(selected_FFT_den(2,:), selected_FFT_den(1,:));
%title('Normal frequency analysis denoised');

%figure(6);
%plot(data_q, c_den);
%title('Cepstrum analysis denoised');
            end
        end
    end
end
end
% save the master_results matrices and the SNR's for each method in an
% external .mat file for processing later
save(output(d, 1:out_leng), 'master_results_env', 'master_results_freq',...
    'master_results_cep', 'SNR_hilbert', 'SNR_normal_freq', 'SNR_cepstrum');
end
```

B.2 The program freq_limit.m

Listing B.2: Function that sets frequency limits in the spectrum.

```
% ************************************************************************
%
%
% *****                          freq_limit
*****
%
% Sets the frequency bandwidth limit
%
% INPUTS: full_spec - full spectrum  - a 2d array, row 1 holds FFT values,
%                     row 2 holds Freq values for FFT bins.
%         bandwidth - the upper cutoff frequency
%
```

```
%
% OUTPUTS: new_spec - new spectrum - the 2d array which has been cut down
% to only include the values relevant to the requested badnwidth
%
% ********************************************************************

function [new_spec] = freq_limit(full_spec, low_BW, up_BW)


% find array indices with freq bins less than up_BW, store in range_low
range_lower = find(full_spec(2,:)<up_BW);

% in range_low find the array of indices with freq bins above low_BW
range_upper = find(full_spec(2,:)>low_BW);

% the last value in range_lower is the bin number of the upper bandwidth
bin_upper = range_lower(1,end);

% the first value in range_upper is the bin number of the lower bandwidth
bin_lower = range_upper(1,1);

%bin_limit = length(range)+1;

% write the new array of FFT values
new_spec(1,:) = full_spec(1,bin_lower:bin_upper);

% write the new array of FFT bin values
new_spec(2,:) = full_spec(2,bin_lower:bin_upper);
```

B.3 The program freq_energy_calc.m

Listing B.3: Function that calculates the energy in a small frequency band.

```
function [freq_energy] = freq_energy_calc(FFT_array, low_lim, up_lim)

% get the frequency band in terms of frequency bin indices
freq_band = FFT_array(2,:);
freq_band(freq_band < low_lim) = 0;
freq_band(freq_band > up_lim) = 0;
low_index_freq = find(freq_band,1);
up_index_freq = find(freq_band,1,'last');

% select the largest component in the frequency band
[M,I] = max(FFT_array(1, low_index_freq:up_index_freq));
peak = FFT_array(2, I+low_index_freq-1);

% select energy band limits around the peak frequency component
local_peak_low = peak - 2;  %2 Hz below
local_peak_up = peak + 2;   %2 Hz above
```

```
% isolate the energy band around the peak component
energy_band = FFT_array(2,:);
energy_band(energy_band < local_peak_low) = 0;
energy_band(energy_band > local_peak_up) = 0;
low_index_energy = find(energy_band,1);
up_index_energy = find(energy_band,1,'last');

% sum the energy in the isolated frequency band
freq_energy = sum(FFT_array(1, low_index_energy:up_index_energy));
```

B.4 The program cep_energy_calc.m

Listing B.4: Function that calculates the energy in a small quefrency band.

```
function [cepstrum_energy] = cep_energy_calc(cep_values,...
    data_q, low_lim, up_lim)

% get the cepstrum band in terms of frequency bin indices
cep_band = data_q;
cep_band(cep_band < low_lim) = 0;
cep_band(cep_band > up_lim) = 0;
low_index_cep = find(cep_band,1);
up_index_cep = find(cep_band,1,'last');

% selecte largest component in the cepstrum band
[M,I] = max(cep_values(low_index_cep:up_index_cep));
peak = data_q(I+low_index_cep-1);

% select energy band limits around the peak cepstrum component
local_peak_low = peak - 0.5;    % 0.35 ms
local_peak_up = peak + 0.5;

% isolate the enrgy banc around the peak component
energy_band = data_q;
energy_band(energy_band < local_peak_low) = 0;
energy_band(energy_band > local_peak_up) = 0;
low_index_energy = find(energy_band,1);
up_index_energy = find(energy_band,1,'last');

% sum the enrgy in the isolated cepstrum band
cepstrum_energy = sum(cep_values(low_index_energy:up_index_energy));
```